ОГЛАВЛЕНИЕ

Л. Левенталь
У. Сэйвилл

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ АССЕМБЛЕРА ДЛЯ МИКРОПРОЦЕССОРОВ 8080 и 8085

Перевод с английского
А. А. Батнера

Москва
«Радио и связь»
1987

ПРЕДИСЛОВИЕ К РУССКОМУ ИЗДАНИЮ

Все возрастающее распространение микропроцессоров и постоянное расширение сферы их применения увеличивают и круг специалистов, занимающихся программированием для них. Большинство здесь составляют программисты, работавшие ранее на машинах с принципиально иными архитектурой и системой команд. Для таких программистов настоящая книга может стать великолепным пособием, помогающим быстро ощутить "дух" программирования для микропроцессоров, выработать иные по сравнению с применявшимися на больших и мини-ЭВМ подходы, тем более, что аналоги микропроцессоров 8080 получили самое широкое распространение в отечественной технике. Сказанное, разумеется, не противоречит тому, что данная книга может быть весьма полезна для всех, кто как-либо связан с программированием для микро-ЭВМ.

Книга состоит из двух частей: обзора вопросов программирования на языке ассемблера и набора подпрограмм. Первая часть хорошо проиллюстрирована большим числом примеров, позволяющих быстро усвоить основные навыки программирования для рассматриваемых микропроцессоров. При этом автор поистине неутомим в повторении некоторых ключевых моментов, добиваясь тем самым их прочного усвоения (это касается, например, порядка представления в ОЗУ 16-разрядных величин). В последней главе этой части (гл. 3) приводятся наиболее распространенные ошибки, встречающиеся при программировании для микропроцессоров 8080 и 8085. Использование материала этой главы может позволить существенно уменьшить время отладки программ за счет исключения типичных ошибок.

Подпрограммы, приведенные во второй части книги, могут удовлетворить потребности большинства программистов в стандартных процедурах. Подпрограммы гл. 10, 11 могут быть весьма полезны для понимания общих принципов организации работы с контроллерами.

Данная книга ориентирована на широкий круг практиков-программистов, но, безусловно, будет полезна также разработчикам кросс-средств для микро-ЭВМ и конструкторам микропроцессорных систем.

А. А. Батнер

ПРЕДИСЛОВИЕ

Эта книга задумана как справочное пособие для программистов, работающих на языке ассемблера. Она содержит краткий обзор вопросов программирования на языке ассемблера для конкретного микропроцессора и набор полезных подпрограмм. В этих подпрограммах использовались стандартные соглашения по формату, документальному оформлению и методам передачи параметров. При этом соблюдались правила наиболее распространенных ассемблеров; кроме того, описаны назначение, процедура, параметры, результаты, время выполнения и требования к памяти.

Для тех, у кого нет времени или необходимости в изучении полного руководства, в обзорных разделах кратко рассмотрено программирование на языке ассемблера. В разделах по программированию эти вопросы обсуждаются более подробно. Глава 1 служит введением в программирование для данного процессора; в ней приводятся основные отличия этого процессора от других микропроцессоров и мини-ЭВМ. В гл. 2 описывается, как выполняются команды и реализуются способы адресации, не доступные в явном виде. В гл. 3 описываются распространенные ошибки программирования.

В наборе подпрограмм делается упор на общие задачи, встречающиеся во многих практических случаях. Эти задачи включают в себя преобразование кодов, обработку массивов, арифметические операции, работу с разрядами, выполнение сдвигов, работу со строками, сортировку и поиск. Приводятся также примеры программ ввода-вывода, обслуживания прерываний и инициализации для распространенного семейства интегральных микросхем, таких как параллельные интерфейсы, последовательные интерфейсы и таймеры. Вы сможете использовать эти программы и подпрограммы непосредственно при решении прикладных задач, а также в более сложных программах в качестве исходного материала.

Эта книга предназначена скорее для тех, кто хочет немедленно использовать язык ассемблера, чем для тех, кто просто желает познакомиться с ним. Читателем может быть:

инженер, техник или программист, который должен писать на языке ассемблера для какого-либо проекта;

пользователь микро-ЭВМ, который хочет писать на языке ассемблера драйверы ввода-вывода, диагностические, вспомогательные или системные программы;

имеющий опыт в программировании на языке ассемблера программист, которому нужно быстро ознакомиться с техническими приемами, применяемыми для данного микропроцессора;

разработчик систем, которому нужны конкретные программы и методы для немедленного использования;

программист, работающий на языке высокого уровня, который должен на уровне ассемблера отлаживать или оптимизировать программы или же связывать программы, написанные на языке высокого уровня, с программами, написанными на языке ассемблера;

программист, занимающийся поддержкой систем, который должен быстро понять, как работает конкретная программа, написанная на языке ассемблера;

владелец микро-ЭВМ, желающий разобраться в ее операционной системе или же модифицировать стандартные программы ввода-вывода или системные программы;

студент, человек, для которого программирование является хобби, или преподаватель, желающие посмотреть примеры работающих программ, написанных на языке ассемблера.

Излагаемый материал может также служить дополнительным пособием для студентов, изучающих программирование на языке ассемблера.

Эта книга призвана сберечь время читателя. У него нет необходимости писать, отлаживать, тестировать или оптимизировать стандартные программы или искать в руководстве по программированию конкретные примеры. Вместо этого читателю предоставляется возможность легко получить определенную информацию, приемы программирования или программы, в которых он нуждается. Чтобы книгой было легче пользоваться, она снабжена указателем.

Совершенно очевидно, что книга, перед которой поставлены подобные цели, требует от читателя обратной связи. Хотя все программы и были полностью проверены и тщательно документированы, однако в случае, если Вы найдете какие-либо ошибки, сообщите, пожалуйста, о них издателю. Если у вас есть предложения по улучшению методов или дополнению материала, программам, если Вы можете дать совет по программированию или рубрикам указателя, сообщите, пожалуйста, о них. При написании этой книги мы использовали наш опыт в программировании, однако в ее улучшении должны помочь Вы. Мы были бы признательны Вам за замечания, критику и предложения.

ТЕРМИНОЛОГИЯ

Для описания архитектуры процессоров 8080 и 8085, обозначения операндов и представления значений чисел и адресов в этой книге использована следующая терминология.

АРХИТЕКТУРА МИКРОПРОЦЕССОРОВ 8080 и 8085

Регистры длиной в байт

A (аккумулятор)
B
C
D
E
H
L
M (адресация ячейки памяти через регистры H и L)
F (флаги)
I (маска прерываний, только в 8085)

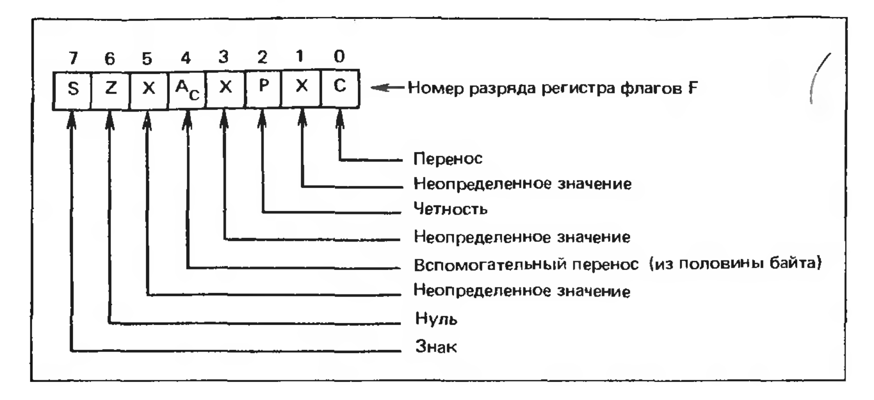


Рис. П.1. Регистр флагов (F)

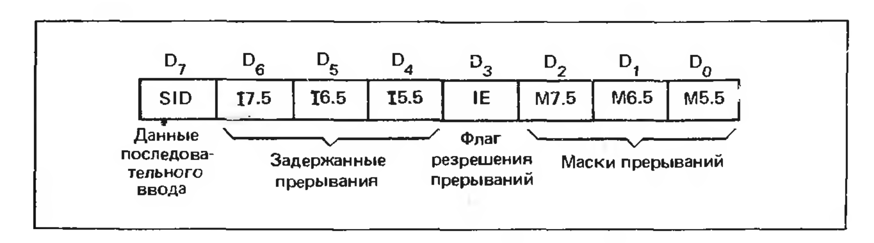


Рис. П.2. Регистр масок прерываний (I) (при чтении командой RIM)

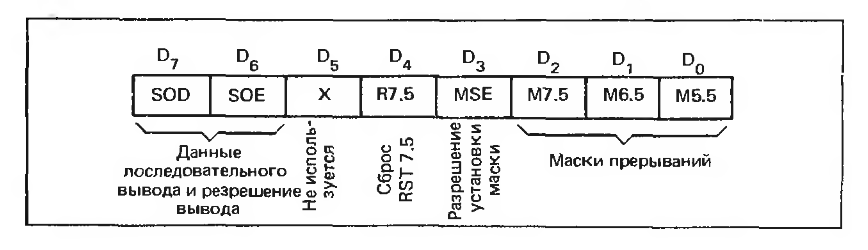


Рис. П.3. Регистр масок прерываний (I) (при записи командой SIM)

Из них регистрами пользователя являются первые семь: A, B, C, D, E, H и L. Регистры F (флаг) и I (маска прерываний) содержат набор разрядов, имеющих независимые функции и смысл. Организация регистра F показана на рис. П.1.

Регистр I (только в 8085) имеет две различные формы: одну при чтении (с помощью команды RIM) и другую при записи (с помощью команды SIM). Организация этих двух вариантов показана на рис. П.2 и П.3.

Пары регистров и регистры длиной в слово

B или BC	(регистры B и C, B – старший байт)
D или DE	(регистры D и E, D – старший байт)
H или HL	(регистры H и L, H – старший байт)
PC	(счетчик команд)
PSW	(слово состояния процессора, аккумулятор и флаги, аккумулятор – старший байт)
SP или S	(указатель стека)

Индивидуальные особенности 8085

IE	Флаг разрешения прерывания
I5.5	Флаг необработанного прерывания для ввода RST 5.5
I6.5	Флаг необработанного прерывания для ввода RST 6.5
I7.5	Флаг необработанного прерывания для ввода RST 7.5
MSE	Разрешение установки маски
M5.5	Разряд маски для ввода RST 5.5
M6.5	Разряд маски для ввода RST 6.5
M7.5	Разряд маски для ввода RST 7.5
Сброс RST 7.5	Разряд используется для сброса триггера RST 7.5
SID	Линия последовательного ввода данных
SOD	Линия последовательного вывода данных
SOE	Разрешение последовательного вывода

Расположение этих разрядов в регистре I показано на рис. П.2, П.3.

Флаги

Вспомогательный перенос из одной половины байта в другую (A_C)
Перенос (C)
Четность (P)
Знак (S)
Ноль (Z)

Расположение этих флагов в регистре F показано на рис. П.1.

АССЕМБЛЕР 8080, 8085

Разделители

:	После метки, за исключением EQU, SET и MACRO, после которых требуется пробел
пробел	После кода команды
,	Между операндами в поле операндов (адресов)
;	Перед комментарием

Псевдооперации

DB	Определить байт; поместить в память данные длиной в байт
DS	Определить область памяти; назначить байты в памяти
DW	Определить слово; поместить в память данные длиной в слово
END	Конец программы
EQU	Приравнять; определить метку псевдооператора
ORG	Определить начало; поместить последующие команды в объектном коде, начиная с заданного адреса

Назначения

Системы счисления:

B (правый индекс)	Двоичная
D (правый индекс)	Десятичная
H (правый индекс)	Шестнадцатеричная
Q (правый индекс)	Восьмеричная

По умолчанию принимается десятичная; шестнадцатеричные числа должны начинаться с цифры (т. е. если число начинается с буквы, то следует в начале добавить нуль).

Другие:

' ' или " "	ASCII (символы окружаются одинарными или двойными кавычками)
¤	Текущее значение счетчика ячеек (счетчика команд)

ОБЩАЯ ТЕРМИНОЛОГИЯ

ADDR	16-разрядный адрес в памяти для данных
BASE	Постоянный 16-разрядный адрес в памяти для данных
BICON	8-разрядный элемент данных в двоичном формате
DEST	16-разрядный адрес в памяти для программы, являющийся адресом назначения для команды перехода
HIGH	16-разрядный элемент данных
INDIR	16-разрядный адрес в памяти для данных, являющийся начальным адресом для косвенного адреса. Косвенный адрес запоминается в ячейках памяти INDIR и INDIR + 1
IPORT	8-разрядный адрес устройства (порта)
LOW	16-разрядный элемент данных
MASK	8-разрядное число, используемое в качестве маски
NTEMP	16-разрядный элемент данных
NTIMES	8-разрядный элемент данных
NTIML	8-разрядный элемент данных
NTIMM	8-разрядный элемент данных
NUM	16-разрядный элемент данных
NUM1	16-разрядный адрес в памяти для данных
NUM2	16-разрядный адрес в памяти для данных
OFFSET	16-разрядный элемент данных
OPER	16-разрядный адрес в памяти для данных
OPER1	16-разрядный адрес в памяти для данных
OPER2	16-разрядный адрес в памяти для данных
OPORT	8-разрядный адрес устройства (порта)
REG	Регистр пользователя (A, B, C, D, E, H или L)
REG1	Другой регистр пользователя, отличающийся от REG
RETPT	16-разрядный адрес в памяти для программы
RP	Пара регистров (B, D или H)
RPH	Старший байт RP
RPL	Младший байт RP
RP1	Другая пара регистров, отличающаяся от RP
RP1H	Старший байт RP1
RP1L	Младший байт RP1
RP2	Другая пара регистров, отличающихся от RP1
RP2H	Старший байт RP2
RP2L	Младший байт RP2
SPTR	16-разрядный адрес в памяти для данных

STRNG	16-разрядный адрес в памяти для данных
SUM	16-разрядный адрес в памяти для данных
VAL16	16-разрядный элемент данных
VAL16H	Старший байт VAL16
VAL16L	Младший байт VAL16
VALUE	8-разрядный элемент данных

ГЛАВА 1

ОБЩИЕ МЕТОДЫ ПРОГРАММИРОВАНИЯ

В этой главе описываются общие методы написания программ на языке ассемблера для микропроцессоров 8080 и 8085. Она содержит способы выполнения следующих операций:

загрузка и сохранение регистров,
запоминание данных в памяти,
арифметические и логические операции,
работа с разрядами,
проверка разрядов,
проверка на определенные значения,
числовые сравнения,
организация циклов (повторяющихся последовательностей операций),
обработка массивов,
поиск в таблице,
работа с символами,
преобразование кодов,
арифметические операции повышенной точности,
умножение и деление,
обработка списков,
обработка структур данных.

В отдельных разделах описываются передача параметров подпрограммам, общие методы написания драйверов ввода-вывода и программ обработки прерываний, а также приемы, позволяющие ускорить выполнение программ и уменьшить используемую ими память.

Описываемые операции необходимы обычно в таких применениях, как контрольно-измерительные приборы, тестовое оборудование, периферийные устройства ЭВМ, аппаратура связи, управление в промышленности, управление процессом, коммерческое оборудование, авиационно-космические и военные системы и производство товаров широкого потребления. Пользователи микро-ЭВМ смогут прибегнуть к этим операциям при написании драйверов ввода-вывода, вспомогательных и диагностических программ и математического обеспечения систем, а также для облегчения понимания, отладки и усовершенствования программ, написанных на языках высокого уровня. Для тех, кто намерен незамедлительно воспользоваться языком ассемблера 8080 и 8085, в данной главе дается краткое руководство по программированию.

1.1. КРАТКИЙ ОБЗОР ДЛЯ ОПЫТНЫХ ПРОГРАММИСТОВ

Для тех, кто знаком с программированием на языке ассемблера на других вычислительных машинах, здесь дается краткий обзор особенностей процессоров 8080 и 8085. Знание этих особенностей поможет сэкономить много времени и труда.

1. Арифметические и логические операции разрешены только между аккумулятором и байтом непосредственных данных или между аккумулятором и регистром общего назначения. Однако один регистр общего назначения в действительности указывает на адрес в памяти; это регистр M, который в действительности обращается к адресу в памяти, содержащемуся в регистрах H и L. Таким образом, команда ADD M, например, означает: *прибавить к аккумулятору содержимое байта памяти, адресуемого через регистры H и L*. Арифметические и логические операции не допускают прямой адресации.

2. Аккумулятор и регистры H и L являются специальными регистрами. Они являются единственными регистрами, которые могут быть прямо загружены или записаны в память. Аккумулятор является единственным регистром, который может быть инвертирован, сдвинут, косвенно загружен с использованием адреса в паре регистров B или D, косвенно записан в память по адресу, содержащемуся в паре регистров B или D, или использован в командах IN и OUT. Регистры H и L составляют единственную пару, которая может быть использована косвенно в арифметических командах, при записи в память данных, заданных непосредственно в команде, или при загрузке и записи в память других регистров, отличных от аккумулятора. Регистры H и L являются также единственной парой, которая может быть передана в счетчик команд или указатель стека. Более того, эти регистры используются как аккумулятор двойной длины при сложении 16-разрядных чисел (команда DAD). Регистры D и E являются в некотором смысле также специальными, поскольку одной командой (XCHG) можно поменять их содержимое с содержимым регистров H и L. Таким образом, регистры в 8080 и 8085 весьма ассимметричны, и программист должен аккуратно выбирать, для каких данных и адресов какими регистрами пользоваться.

3. Часто для одних и тех же физических регистров используется несколько имен. Для многих команд A, B, C, D, E, H и L являются 8-разрядными регистрами. Для других команд регистры B и C (B – старший по значению), D и E (D – старший по значению) или H и L (H – старший по значению) являются 16-разрядной парой регистров. Термины *пара регистров B*, *регистры B и C* и *пара регистров BC* имеют одно и то же значение; подобные же варианты существуют для регистров D и E и H и L. Заметим, что пара регистров и два одиночных регистра физически одно и то же, и они не могут служить одновременно для различных целей.

Регистры H и L фактически почти всегда применяют для косвенного адреса из-за наличия команд, имеющих доступ к регистру M, и таких специальных команд, как SPHL, PCHL, XTHL и XCHG. Благодаря тому, что существует команда XCHG, для второго адреса берут регистры D и E, а не B и C. Регистры B и C используют обычно как отдельные 8-разрядные регистры для временного хранения данных.

4. Воздействие различных команд на флаги весьма непоследовательно. К некоторым особенно необычным действиям относятся следующие: а) ло-

гические команды очищают флаг переноса, б) команды сдвига не действуют на другие флаги, кроме флага переноса, в) команды загрузки, записи, пересылки, увеличения на 1 пары регистров и уменьшения на 1 пары регистров вообще не оказывают влияния на флаги, г) 16-разрядное сложение действует только на флаг переноса. Определить, как команды действуют на флаги, можно с помощью табл. А.1 (приложение А).

5. Отсутствуют косвенная адресация через память и индексация. Отсутствие косвенной адресации через память компенсируется загрузкой косвенного адреса в регистры H и L. Действительная косвенная адресация, таким образом, является двухшаговым процессом. При желании загрузить или записать в память аккумулятор можно также загрузить косвенный адрес в регистры B и C или D и E.

Отсутствие индексной адресации компенсируется добавлением пары регистров с помощью команды DAD. Эта команда добавляет пару регистров к H и L. Таким образом, индексация требует нескольких шагов: а) загрузить индекс в пару регистров, б) загрузить базовый адрес в другую пару (одной из пар регистров должны быть H и L), в) используя команду DAD, сложить две пары и г) использовать сумму как косвенный адрес (при помощи обращения к регистру M). Индексация в 8080 и 8085 – долгий и неудобный процесс.

6. Нет флага переполнения при получении дополнения до двух, так что надо определять такое переполнение программным путем. Из этого следует, что трудно работать с числами со знаком.

7. Многие обычные команды отсутствуют, но могут быть легко смоделированы с помощью регистровых команд. Примерами являются очистка аккумулятора (с использованием SUB A или XRA A), логический сдвиг аккумулятора влево (с помощью ADD A), очистка флага переноса (ANA A или ORA A) и проверка аккумулятора (ANA A или ORA A). Команды ANA A и ORA A очищают флаг переноса и устанавливают остальные флаги в соответствии с содержимым аккумулятора. Напомним, что загрузка регистра не действует на флаги.

8. Нет относительных переходов. Фактически, единственной командой перехода, которая не требует абсолютного адреса, является PCHL, по которой загружается счетчик команд из регистров H и L и, таким образом, производится косвенный переход.

9. Есть два отдельных набора команд увеличения и уменьшения на 1, Команды DCR и INR применяются к 8-разрядным регистрам и действуют на все флаги, за исключением флага переноса. Команды DCX и INX применяются к 16-разрядным парам регистров и вообще не действуют на флаги. Вы можете использовать 16-разрядные пары регистров как обыкновенные счетчики, но единственным способом проверки пары на 0 является использование команды логическое ИЛИ к двум регистрам вместе с аккумулятором.

10. Нет арифметических или логических сдвигов. Единственными командами сдвига являются команды циклического сдвига с флагом переноса или без него. Другие сдвиги могут быть смоделированы при помощи команд циклического сдвига (RRC, RLC, RAR и RAL) и команд сложения (ADD A, ADC A и DAD H). Флаг переноса может быть установлен с помощью STC, а очищен с помощью ANA A (или ORA A).

11. Аккумулятор является единственным регистром, который может быть сдвинут, инвертирован или использован для ввода или вывода. Единственными командами, которые оперируют непосредственно с регистрами общего назначения, являются команды MOV (пересылка содержимого в другой регистр или из другого регистра), MVI (загрузка непосредственного операнда), DCR (уменьшение на 1) и INR (увеличение на 1). Эти команды могут оперировать также с регистром M, т. е. байтом из памяти, адресуемым через регистры H и L.

12. В стек или из стека могут быть переданы только пары регистров. Одной из таких пар является слово состояния процессора (PSW), которое содержит аккумулятор (старший байт) и флаги (младший байт). Команды CALL и RETURN передают адреса в стек или из него.

13. В микропроцессоре 8080 отсутствует читаемый флаг системы прерываний. Это создает трудности в том случае, когда исходное состояние системы прерываний должно быть восставновлено после выполнения секции команд, которая должна выполняться при закрытых прерываниях. Для решения этой проблемы можно копию состояния прерываний хранить в ОЗУ. С другой стороны, 8085 имеет читаемый флаг разрешения прерываний.

14. В микропроцессорах 8080 и 8085 приняты следующие общие соглашения.

• При записи всех 16-разрядных адресов младший байт записывается первым (т. е. по меньшему адресу). Порядок байтов в адресах тот же, что и в микропроцессорах Z80 и 6502, но является обратным порядку байтов, принятому в микропроцессорах 6800 и 6809.

• Указатель стека содержит младший адрес, действительно занятый в стеке. Это соглашение также принято в микропроцессорах Z80 и 6809, но явно противоположно принятому в 6502 и 6800 (следующий доступный адрес). Согласно всем командам 8080 и 8085 данные в стек записываются с предварительным уменьшением на 1 (вычитанием перед записью байта 1 из указателя стека) и загружаются из стека с последующим увеличением на 1 (добавлением после загрузки байта 1 к указателю стека).

• Флаг разрешения прерываний (только в 8085), равный 1, разрешает прерывания, а 0 — запрещает их. Такое же соглашение принято и в Z80, но оно обратно принятому в 6502, 6800 и 6809.

1.2. НАБОР РЕГИСТРОВ

Программирование на языке ассемблера 8080/8085 осложняется ассиметричностью системы команд этих процессоров. Многие команды применяются только к определенным регистрам, парам регистров или наборам регистров. Почти каждый регистр обладает индивидуальными особенностями, и почти каждая команда имеет свою специфику. В табл. 1.1 перечислены регистры длиной в байт и команды, которые оперируют с ними. В табл. 1.2 перечислены пары регистров (или регистры длиной в слово) и оперирующие с ними команды (все команды, конечно, неявно изменяют программный счетчик). В табл. 1.3 перечислены содержащиеся в парах регистров косвенные адреса и команды, в которых используются эти адреса. В табл. 1.4 перечислены команды, применимые только к аккумулятору, а в табл. 1.5 — команды,

применимые только к определенным парам регистров. В табл. 1.6 перечислены команды, относящиеся к стеку.

Таблица 1.1. *8-разрядные регистры и применяемые команды*

Регистр	Команды
A	ACI, ADC, ADD, ADI, ANA, ANI, CMA, CMP, CPI, DAA, DCR, IN, INR, LDA, LDAX, MOV, MVI, ORA, ORI, OUT, RAL, RAR, RIM (только в 8085), RLC, RRC, SBB, SBI, SIM (только в 8085), STA, STAX, SUB, SUI, XRA, XRI
B, C, D, E, H, L	ADC, ADD, ANA, CMP, DCR, INR, MOV, MVI, ORA, SBB SUB, XRA
F (флаги)	CMC, STC (см. также пару регистров PSW)
I (маска прерываний, только в 8085)	RIM, SIM (только в 8085)

Таблица 1.2. *Пары регистров и применяемые команды*

Пара регистров	Команды
B (B и C)	DAD, DCX, INX, LXI, POP, PUSH,
D (D и E)	DAD, DCX, INX, LXI, POP, PUSH, XCHG
H (H и L)	DAD, DCX, INX, LHLD, LXI, PCHL, POP, PUSH, SHLD, SPHL, XCHG, XTHL
PSW (A и флаги)	POP, PUSH
Счетчик команд	Команды вызова подпрограммы, команды перехода, команды возврата из подпрограммы, RST
Указатель стека	Команды вызова подпрограммы, DAD, DCX, INX, LXI, POP, PUSH, команды возврата из подпрограммы, RST, SPHL

Таблица 1.3. *Косвенные адреса и применяемые команды*

Расположение адреса	Команды
Пара регистров B (B и C)	LDAX, STAX
Пара регистров D (D и E)	LDAX, STAX
Пара регистров H (H и L)	ADC, ADD, ANA, CMP, DCR, INR, MOV, MVI, ORA, SBB, SUB, XRA
Указатель стека	Команды вызова подпрограммы, POP, PUSH, команды возврата из подпрограммы, RST, XTHL

Таблица 1.4. *Команды, которые применяются только к аккумулятору*

Команда	Функция	Команда	Функция
ACI	Непосредственное сложение с переносом	RAR	Циклический сдвиг вправо через перенос

Команда	Функция	Команда	Функция
ADI	Непосредственное сложение	RIM	Чтение маски прерываний (только в 8085)
ANI	Непосредственное логическое И	RLC	Циклический сдвиг влево
CMA	Дополнение (логическое)	RRC	Циклический сдвиг вправо
CPI	Непосредственное сравнение	SBI	Непосредственное вычитание с заемом
DAA	Десятичная коррекция		
IN	Ввод	SIM	Установка маски прерываний (только в 8085)
LDA	Прямая загрузка		
LDAX	Косвенная загрузка	STA	Прямое запоминание
ORI	Непосредственное логическое ИЛИ	STAX	Косвенное запоминание
		SUI	Непосредственное вычитание
OUT	Вывод		
RAL	Циклический сдвиг влево через перенос	XRI	Непосредственное логическое ИСКЛЮЧАЮЩЕЕ ИЛИ

Таблица 1.5. *Команды, которые применяются только к одной или двум парам регистров*

Команда	Пары регистров	Функция
LDAX	В или D	Косвенная загрузка аккумулятора
LHLD	H	Прямая загрузка H и L
PCHL	H, PC	Пересылка H и L в PC
SHLD	H	Прямая запись в память H и L
STAX	В или D	Косвенная запись в память аккумулятора
XCHG	D, H	Обмен HL с DE
XTHL	H	Обмен HL с вершиной стека

Таблица 1.6. *Команды, в которых используется стек*

Команда	Функция
Команды вызова подпрограммы	Переход и сохранение счетчика команд в стеке
POP	Загрузка пары регистров из стека
PUSH	Запись пары регистров в стеке
Команды возврата из подпрограммы	Загрузка счетчика команд из стека
RST	Переход по адресу вектора с сохранением счетчика команд в стеке
XTHL	Обмен H и L с вершиной стека

Регистры обычно используются следующим образом.

• Аккумулятор является центром обработки данных; для большинства арифметических и логических команд он служит источником одного из операндов и местом назначения для результата.

• Регистры H и L (пара регистров H) представляют собой основной регистр адреса памяти. Команды, которые ссылаются на регистр M, в действительности ссылаются на адрес памяти в этой паре регистров.

• Регистры D и E (пара регистров D) представляют собой дополнительный регистр адреса памяти, так как программист может поменять их содержимое с содержимым H и L, используя команду XCHG.

• Регистры B и C (лара регистров B) являются регистрами общего назначения без каких-либо отличительных особенностей, хотя команды LDAX (загрузить аккумулятор косвенно) и STAX (запомнить аккумулятор косвенно) могут использовать их в качестве адресных регистров. Обычно программисты используют регистры B и C для счетчиков и временного хранения данных.

Индивидуальные особенности отдельных регистров можно описать следующим образом:

Аккумулятор. Единственный регистр длиной в байт, который может быть прямо загружен или записан в память. Единственный регистр, который можно сдвигать, инвертировать или корректировать в двоично-десятичный вид с помощью одной команды. Единственный регистр, который может быть записан в порт вывода (с помощью команды OUT) или загружен из порта вывода (с помощью команды IN). Исходный регистр и регистр назначения для всех арифметических и логических команд, за исключением DAD, DCR, DCX, INR и INX. Единственный регистр, который может быть загружен из регистра маски прерываний (командой RIM) или записан в регистр маски прерываний (SIM). (Регистром маски прерываний снабжен только процессор 8085.)

Регистры H и L. Единственная пара регистров, которая может быть использована косвенно (с помощью ссылки на регистр M) в командах ADC, ADD, ANA, CMP, DCR, INR, MOV, MVI, ORA, SBB, SUB и XRA. Единственная пара регистров, которая может быть прямо загружена или записана в память. Исходные регистры и регистры назначения для команды DAD. Единственная пара регистров, которую можно обменять с регистрами D и E и с вершиной стека. Единственная пара регистров, которую можно переслать в указатель стека (используя команду SPHL) или в счетчик команд (командой PCHL). Единственная пара регистров, которая может быть сдвинута одной командой (DAD H).

Регистры D и E. Единственная пара регистров, для которой возможен обмен данными с регистрами H и L (с помощью команды XCHG).

Указатель стека. Единственный адресный регистр, который обеспечивает автоприращение (приращение адреса после выполнения команды) и автоуменьшение (уменьшение адреса до выполнения команды). Может быть загружен только командой LXI или SPHL. Его значение можно определить только с помощью загрузки 0 в пару регистров H и L с последующим использованием команды DAD SP. Единственная пара регистров, которая может быть использована для передачи других пар регистров в память или из памяти (команды PUSH и POP) или для записи в память или чтения из нее счетчика команд (команды CALL и RETURN).

Слово состояния процессора (PSW). Содержит в себе аккумулятор (старший байт) и флаги (младший байт). Может быть только передано в стек или из стека с помощью команд PUSH и POP.

Отметим следующее:

• Только аккумулятор и регистры H и L могут быть прямо загружены из памяти или записаны в память. Для других регистров или пар регистров нет команд, эквивалентных командам LDA, STA, LHLD или SHLD.

• Только адрес в H и L (регистр M) можно использовать для всех команд, кроме загрузки или записи аккумулятора в память. Только адрес в H и L может быть использован для передачи данных в другие регистры или из них в арифметических или логических командах.

• Только команды DCR, DCX, DAD, INR и INX позволяют выполнять арифметические операции без использования аккумулятора (команды DCR и INR можно применять и для аккумулятора). Из этих команд только DCR, DAD и INR изменяют значение флагов; DCR и INR изменяют все флаги, кроме переноса, в то время как DAD изменяет только флаг переноса.

1.2.1. ПЕРЕДАЧА ИЗ РЕГИСТРА В РЕГИСТР

Команда MOV может передавать любой 8-разрядный регистр общего назначения (A, B, C, D, E, H или L) в любой другой 8-разрядный регистр общего назначения. Регистр флагов (F) может быть только передан в стек или получен из стека вместе с аккумулятором (с помощью PUSH PSW и POP PSW). Регистр маски прерываний (только в 8085) может быть передан в аккумулятор или из него с помощью команд RIM или SIM. Команда XCHG обменивает пары регистров D и H.

Обычно используют следующие команды передачи:

MOV A, REG передает содержимое регистра в аккумулятор;
MOV REG, A передает содержимое аккумулятора в регистр;
MOV REG, M загружает регистр содержимым памяти по адресу в регистрах H и L;
MOV M, REG записывает содержимое регистра в память по адресу в регистрах H и L;
XCHG обменивает содержимое пары регистров D (регистры D и E) с содержимым пары регистров H (регистры H и L)

В команде MOV регистр назначения является первым операндом, таким образом, MOV REG1, REG2 передает содержимое REG2 в REG1, что обратно соглашению, предложенному в стандарте 694 IEEE для команд языка ассемблера [1, 2]. Команда MOV изменяет регистр назначения, но не влияет на регистр-источник. Команда XCHG изменяет четыре регистра (D, E, H и L); таким образом, эта команда эквивалентна четырем командам MOV плюс некоторым промежуточным командам, сохраняющим один байт данных во время передачи другого. Ни MOV, ни XCHG не влияют на флаги.

1.3. ЗАГРУЗКА РЕГИСТРОВ ИЗ ПАМЯТИ

В микропроцессорах 8080 и 8085 предусмотрены четыре способа адресации, которыми можно пользоваться при загрузке регистров из памяти: прямая (из памяти с конкретным адресом), непосредственная (с конкретным значением), косвенная (из адреса, помещенного в паре регистров) и стековая (из вершины стека) [3].

1.3.1. ПРЯМАЯ ЗАГРУЗКА РЕГИСТРОВ

С использованием прямой адресации из памяти могут быть загружены только аккумулятор и регистры H и L (пара регистров H).

Примеры

1. **LDA 2050H**

Эта команда загружает аккумулятор (регистр A) из ячейки памяти 2050_{16}.

2. **LHLD 0A00H**

Эта команда загружает регистр L из ячейки памяти $A000_{16}$, а регистр H из ячейки памяти $A001_{16}$. Заметим, что по принятому для 8080, 8085 формату хранения 16-разрядных чисел первым является младший по значению байт, а за ним в ячейке со следующим адресом – старший по значению байт.

1.3.2. НЕПОСРЕДСТВЕННАЯ ЗАГРУЗКА РЕГИСТРОВ

Непосредственная адресация может быть использована для загрузки любого регистра или пары регистров, причем последние включают в себя и указатель стека.

Примеры

1. **MVI C,6**

Эта команда загружает регистр C значением 6. Здесь 6 является 8-разрядным числом, а не 16-разрядным адресом; не следует смешивать число 6 с адресом 0006_{16}.

2. **LXI D,15E3H**

Эта команда загружает 15_{16} в регистр D и E 3_{16} в регистр E.

1.3.3. КОСВЕННАЯ ЗАГРУЗКА РЕГИСТРОВ

Команда MOV REG,M может загрузить любой регистр из ячейки памяти, адрес которой содержится в регистрах H и L. Команда LDAX может загрузить аккумулятор с использованием адреса, содержащегося в паре регистров B или D. Заметим, что нет команды, загружающей косвенно пару регистров.

Примеры

1. **MOV D,M**

Эта команда загружает регистр D из ячейки памяти, адрес которой содержится в регистрах H и L. Команда языка ассемблера имеет форму: MOV регистр назначения, исходный регистр; этот порядок регистров является обратным, предлагаемому в стандарте 694 IEEE [1].

2. **LDAX B**

Эта команда загружает аккумулятор из ячейки памяти, адрес которой содержится в регистрах B и C. Команда MOV A,M имеет то же самое назначение, но в ней используется адрес, содержащийся в регистрах H и L. Заметим, однако, что нельзя использовать B и C или D и E для косвенной загрузки любого регистра, кроме аккумулятора.

1.3.4. ЗАГРУЗКА РЕГИСТРОВ ИЗ СТЕКА

Команда POP RP загружает пару регистров из вершины стека и соответственно устанавливает указатель стека. Одной из пар регистров для команды POP является слово состояния процессора (PSW), в котором содержится аккумулятор (старший байт) и флаги (младший байт). Не существует команд, загружающих один регистр из стека или использующих указатель стека косвенно без его изменения (хотя команда XTHL в результате и не оказывает влияния на указатель стека, но она передает данные как в стек, так и из стека).

Пример

POP D

Эта команда загружает регистры D и E из вершины стека и увеличивает указатель стека на 2. Регистр E загружается первым в соответствии с форматом для 16-разрядных чисел, принятым в 8080, 8085.

Стек имеет следующие характерные особенности.

• Указатель стека содержит адрес ячейки, которая была занята самой последней (младший занятый адрес). Стек может быть расположен в любом месте памяти.

• Данные запоминаются в стеке с использованием предуменьшения, т. е. команды уменьшают указатель стека на 1 перед запоминанием каждого байта. Данные загружаются из стека с использованием послеувеличения, т. е. команды увеличивают указатель стека на 1 после загрузки каждого байта.

• Как это типично для микропроцессоров, отсутствуют указатели выхода за границы стека в ту или иную сторону.

1.4. ЗАПОМИНАНИЕ РЕГИСТРОВ В ПАМЯТИ

Для запоминания регистров в памяти существуют три способа адресации: прямая (в память с конкретным адресом), косвенная (в память с адресом, который находится в паре регистров) и стековая (в вершину стека).

1.4.1. ПРЯМОЕ ЗАПОМИНАНИЕ РЕГИСТРОВ

Прямая адресация может быть использована только для запоминания аккумулятора или регистров H и L.

Примеры

1. **STA 35C8H**

При выполнении этой команды запоминается аккумулятор в ячейке памяти $35C8_{16}$.

2. **SHLD 203AH**

При выполнении этой команды запоминается регистр L в ячейке памяти $203A_{16}$, а регистр H – в ячейке памяти $203B_{16}$, т. е, как обычно, в обратном порядке.

1.4.2. КОСВЕННОЕ ЗАПОМИНАНИЕ РЕГИСТРОВ

При выполнении этой команды MOV M,REG может запоминаться любой регистр по адресу, который находится в регистрах H и L. Команда STAX может запомнить аккумулятор по адресу, который находится в паре регистров B или D. Заметим, что нет команды для косвенного запоминания пары регистров.

Примеры

1. **MOV M,C**

При выполнении этой команды запоминается регистр C по адресу, который содержится в регистрах H и L. Команда формируется в виде: *переслать в M из C.*

2. **STAX D**

При выполнении этой команды запоминается аккумулятор в памяти по

адресу, содержащемуся в регистрах D и E. Команда MOV M,A имеет то же самое назначение, но в ней используется адрес в регистрах H и L. Заметим, однако, что аккумулятор является единственным регистром, который можно запомнить косвенно с помощью регистров D и E или B и C.

1.4.3. ЗАПОМИНАНИЕ РЕГИСТРОВ В СТЕКЕ

При выполнении команды PUSH RP запоминается пара регистров в вершине стека и устанавливается соответственно указатель стека. Одной из пар регистров является слово состояния процессора (PSW), которое содержит аккумулятор (старший байт) и флаги (младший байт). Нет команды, при выполнении которой запоминается в стеке один регистр.

Пример

PUSH B

При выполнении этой команды запоминаются регистры B и C в вершине стека и указатель стека уменьшается на 2. Регистр B запоминается первым, поэтому C заканчивает стек в его вершине.

1.5. ДРУГИЕ ОПЕРАЦИИ ЗАГРУЗКИ И ЗАПОМИНАНИЯ

Другие операции загрузки и запоминания требуют не одну, а большее число команд. Типичные примеры таких операций:

1. Прямая загрузка любого регистра, отличного от A:

```
        LDA        ADDR
        MOV        REG,A
```

или

```
        LXI        H,ADDR
        MOV        REG,M
```

При втором методе A остается без изменения, но используются H и L. Само собой разумеется, адрес в H и L может пригодиться для последующего использования.

2. Косвенная загрузка любого регистра (из памяти, адрес которой содержится в ячейках INDIR и INDIR + 1):

```
        LHLD       INDIR      ;ВЗЯТЬ КОСВЕННЫЙ АДРЕС
        MOV        REG,M      ;ЗАГРУЗИТЬ ДАННЫЕ КОСВЕННО
```

3. Прямая загрузка любой пары регистров, отличной от H и L:

регистров D и E

```
        LHLD       ADDR       ;ПЕРЕСЛАТЬ ДАННЫЕ В HL
        XCHG                  ;И ЗАТЕМ В DE
```

Команда XCHG служит специально для обмена пары регистров D с парой регистров H:

регистров B и C

```
        LHLD       ADDR       ;ПЕРЕСЛАТЬ ДАННЫЕ В HL
        MOV        B,H        ;И ЗАТЕМ В BC ПО БАЙТУ ЗА ОДИН РАЗ
        MOV        C,L
```

указателя стека

```
        LHLD       ADDR       ;ПЕРЕСЛАТЬ ДАННЫЕ В HL
        SPHL                  ;И ЗАТЕМ В SP
```

Команда SPHL служит для передачи H и L в указатель стека.

4. Прямое запоминание любых регистров, отличных от A:

```
        MOV     A,REG
        STA     ADDR
```

или

```
        LXI     H,ADDR
        MOV     M,REG
```

5. Косвенное запоминание любого регистра (в памяти, адрес которой содержится в ячейках INDIR и INDIR + 1):

```
        LHLD    INDIR   ;ВЗЯТЬ КОСВЕННЫЙ АДРЕС
        MOV     M,REG   ;ЗАПОМНИТЬ ДАННЫЕ ПО ЭТОМУ АДРЕСУ
```

6. Прямое запоминание любой пары регистров, отличной от H и L: регистров D и E

```
        XCHG            ;ПЕРЕСЛАТЬ DE В HL
        SHLD    ADDR    ;И ЗАТЕМ В ПАМЯТЬ
```

регистров B и C

```
        MOV     H,B     ;ПЕРЕСЛАТЬ BC В HL ПО БАЙТУ ЗА ОДИН РАЗ
        MOV     L,C
        SHLD    ADDR    ;И ЗАТЕМ В ПАМЯТЬ
```

указателя стека

```
        LXI     H,0     ;ПЕРЕСЛАТЬ SP В HL
        DAD     SP
        SHLD    ADDR    ;И ЗАТЕМ В ПАМЯТЬ
```

Не существует команды, аналогичной SPHL, для пересылки данных в противоположном направлении.

1.6. ЗАПОМИНАНИЕ ДАННЫХ В ОЗУ

Начальные значения ячеек ОЗУ задаются либо через аккумулятор, либо прямо или косвенно с использованием регистров H и L.

Примеры

1. Запомнить 8-разрядный элемент (VALUE) по адресу ADDR:

```
        MVI     A,VALUE
        STA     ADDR
```

или

```
        LXI     H,ADDR
        MVI     M,VALUE
```

В первом случае, если VALUE = 0, MVI A, VALUE можно заменить на SUB A или на XRA A. Заметим, что SUB A или XRA A изменяют флаги, в то время как MVI A, 0 не изменяет.

2. Запомнить 16-разрядный элемент (VAL16) по адресам ADDR и ADDR+1 (старшим по значению байтом является ADDR + 1):

```
        LXI     H,VAL16
        SHLD    ADDR
```

3. Запомнить 8-разрядный элемент (VALUE) по адресу, содержащемуся в ячейках памяти INDIR и INDIR + 1:

```
        LHLD    INDIR   ;ВЗЯТЬ КОСВЕННЫЙ АДРЕС
        MVI     M,VALUE ;ЗАПОМНИТЬ ЗНАЧЕНИЕ КОСВЕННО
```

1.7. АРИФМЕТИЧЕСКИЕ И ЛОГИЧЕСКИЕ ОПЕРАЦИИ

Для большинства арифметических и логических операций (сложение, вычитание, логическое И, логическое ИЛИ, ИСКЛЮЧАЮЩЕЕ ИЛИ и сравнение) одним из операндов является аккумулятор, а вторым 8-разрядный регистр или байт данных, заданный непосредственно в команде. Результат (если он существует) помещается в аккумулятор. Если используется регистр М, то процессор получает операнд из памяти по адресу, который содержится в регистрах Н и L.

Примеры

1. Добавить регистр В к аккумулятору

```
        ADD     B
```

Сумма остается в аккумуляторе.

2. Выполнить операцию логическое И для аккумулятора и двоичного значения BICON:

```
        ANI     BICON
```

Непосредственная адресация требует специального кода операции.

3. Выполнить операцию логическое ИЛИ для аккумулятора и числа, адрес которого содержится в регистрах Н и L:

```
        ORA     M
```

Регистр М в действительности ссылается на адрес, содержащийся в Н и L.

Другие операции требуют больше одной команды. Здесь приводятся типичные примеры таких операций.

- Сложить содержимое ячеек памяти OPER1 и OPER2, сумму поместить в SUM

```
        LDA     OPER1      ;ПОЛУЧИТЬ ПЕРВЫЙ ОПЕРАНД
        MOV     B,A
        LDA     OPER2      ;ПОЛУЧИТЬ ВТОРОЙ ОПЕРАНД
        ADD     B
        STA     SUM        ;СОХРАНИТЬ СУММУ
```

или

```
        LXI     H,OPER1    ;ПОЛУЧИТЬ ПЕРВЫЙ ОПЕРАНД
        MOV     A,M
        LXI     H,OPER2    ;ПОЛУЧИТЬ ВТОРОЙ ОПЕРАНД
        ADD     M
        LXI     H,SUM      ;СОХРАНИТЬ СУММУ
        MOV     M,A
```

Второй вариант можно существенно сократить, если операнды и сумма имеют последовательные адреса памяти. Например, если OPER2 = OPER1 + 1 и SUM = OPER2 + 1, то в результате будет

```
        LXI     H,OPER1
        MOV     A,M        ;ПОЛУЧИТЬ ПЕРВЫЙ ОПЕРАНД
        INX     H
        ADD     M          ;ДОБАВИТЬ ВТОРОЙ ОПЕРАНД
        INX     H
        MOV     M,A        ;СОХРАНИТЬ СУММУ
```

• Добавить константу (VALUE) к ячейке памяти OPER

```
        LDA     OPER
        ADI     VALUE
        STA     OPER
```

или

```
        LXI     H,OPER
        MOV     A,M
        ADI     VALUE
        MOV     M,A
```

Если VALUE = 1 или −1, то последние три команды можно заменить на INR M или DCR M. Ни та, ни другая команда не изменяют аккумулятор.

1.8. РАБОТА С РАЗРЯДАМИ

Программист может установить, очистить, получить обратный код (дополнение к 1) или проверить разряды, используя логические операции с соответствующими масками. Команды сдвига и получения обратного кода могут оперировать только с аккумулятором, но в то же время для выполнения небольшого числа сдвигов могут использоваться арифметические и логические команды. Дополнительные примеры работы с разрядами содержатся в гл. 7.

Возможны следующие операции с отдельными разрядами аккумулятора:

установить с помощью операции логическое ИЛИ с единицами в соответствующих позициях;

очистить с помощью операции логическое И с нулями в соответствующих позициях;

инвертировать (изменить на обратное значение) с помощью операции ИСКЛЮЧАЮЩЕЕ ИЛИ с единицами в соответствующих позициях;

проверить (на все нули в проверяемых разрядах) с помощью операции логическое И с единицами в соответствующих позициях.

Примеры

1. Установить разряд 6 аккумулятора:

```
    ORI  01000000B ;УСТАНОВИТЬ РАЗРЯД 6 С ПОМОЩЬЮ "ИЛИ" С 1
```

Операция логическое ИЛИ разряда с нулем оставляет разряд без изменения.

2. Очистить разряд 3 аккумулятора:

```
    ANI  11110111B ;ОЧИСТИТЬ РАЗРЯД 3 С ПОМОЩЬЮ "И" С 0
```

Операция логическое И разряда с единицей оставляет разряд без изменения.

3. Инвертировать (изменить на обратное значение) разряд 2 аккумулятора:

```
    XRI  00000100B ;ИНВЕРТИРОВАТЬ РАЗРЯД 2 С ПОМОЩЬЮ
                   ; "ИСКЛЮЧАЮЩЕГО ИЛИ" С 1
```

Операция ИСКЛЮЧАЮЩЕЕ ИЛИ разряда с нулем оставляет разряд без изменения.

4. Проверить разряд 5 аккумулятора. Очистить флаг нуля, если разряд 5 равен 1, и установить флаг нуля, если разряд 5 равен 0:

```
    ANI  00100000B ;ПРОВЕРИТЬ РАЗРЯД 5 С ПОМОЩЬЮ "И" С 1
```

Отметим инверсию в данном случае; флаг нуля устанавливается в 1, если разряд равен 0.

Используя соответствующую маску, можно изменить за один раз больше одного разряда.

5. Установить разряды 4 и 5 аккумулятора:

```
ORI  00110000B ;УСТАНОВИТЬ РАЗРЯДЫ 4 И 5 С ПОМОЩЬЮ "ИЛИ" С 1
```

6. Инвертировать (изменить на обратное значение) разряды 0 и 7 аккумулятора:

```
XRI  10000001B ;ИНВЕРТИРОВАТЬ РАЗРЯДЫ 0 И 7 С ПОМОЩЬЮ
               ; "ИСКЛЮЧАЮЩЕГО ИЛИ" С 1
```

Единственный способ работы с разрядами в других регистрах или в памяти состоит в пересылке значения в аккумулятор.

• Установить разряд 4 регистра С:

```
MOV  A,C
ORI  00010000B
MOV  C,A
```

• Очистить разряд 1 в ячейке памяти ADDR

```
LDA  ADDR
ANI  11111101B
STA  ADDR
```

Иногда рационально работать с разрядом 0 какого-либо регистра или ячейки памяти с использованием команды INR или DCR. Любая из этих команд изменяет значение разряда 0 на противоположное; т. е. как INR, так и DCR устанавливают разряд 0, если он был равен 0, или очищают его, если он был равен 1. Этот рациональный способ полезен в том случае, когда заданный регистр или ячейка памяти содержит только один 1-разрядный флаг.

Команды RAL и RAR циклически сдвигают аккумулятор вместе с флагом переноса так, как если бы они составляли 9-разрядный регистр. На рис. 1.1 и 1.2 показано действие команд RAL и RAR. Как видно из рис. 1.3 и 1.4, команды RLC и RRC циклически сдвигают один аккумулятор; разряд, выдвигаемый из конца аккумулятора, появляется как во флаге переноса, так и в разряде с другого конца аккумулятора. Сложение аккумулятора с самим собой, как показано далее, заменяет другие команды сдвига:

ADD A сдвигает логически аккумулятор на одну позицию влево, как показано на рис. 1.5; при этом разряд 0 очищается;

Рис. 1.1. Команда RAL (циклически сдвигать аккумулятор влево через перенос)

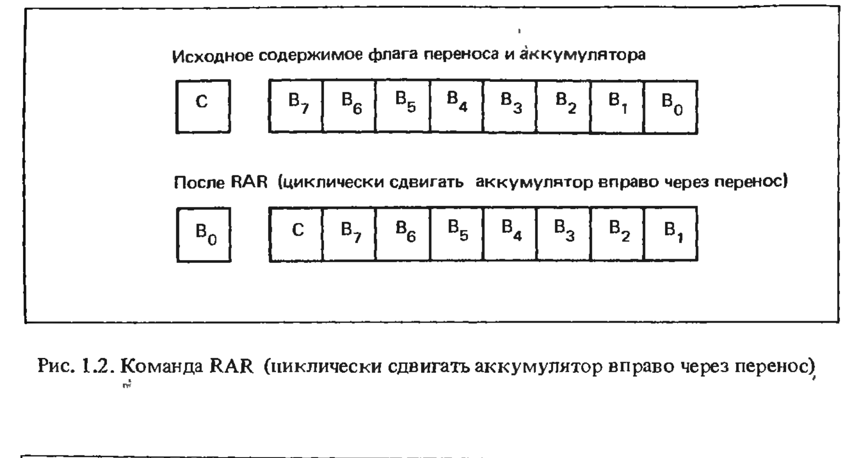


Рис. 1.2. Команда RAR (циклически сдвигать аккумулятор вправо через перенос)

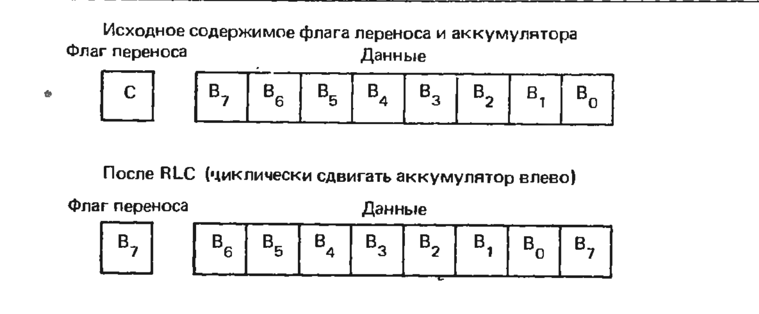


Рис. 1.3. Команда RLC (циклически сдвигать аккумулятор влево)

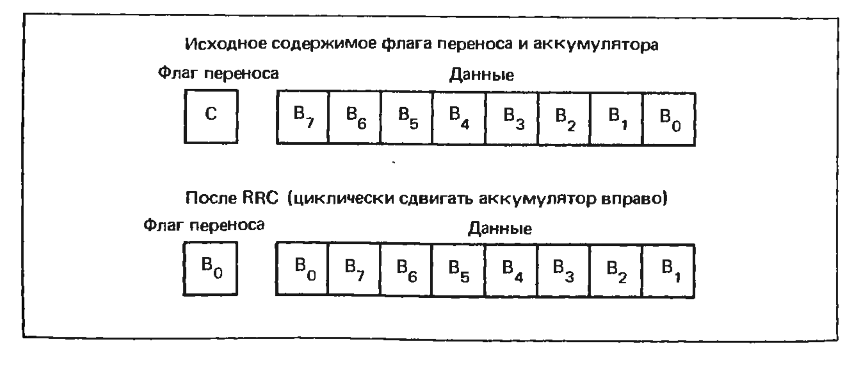


Рис. 1.4. Команда RRC (циклически сдвигать аккумулятор вправо)

Исходное содержимое флага переноса и аккумулятора

C		B7	B6	B5	B4	B3	B2	B1	B0

После ADD A (сложить аккумулятор с самим собой)

B7		B6	B5	B4	B3	B2	B1	B0	0

Рис. 1.5. Команда ADD A (сложить аккумулятор с самим собой)

ADC A циклически сдвигает аккумулятор и флаг переноса влево на одну позицию; ADC A отличается от RAL только тем, что воздействует на все флаги, в то время как RAL изменяет только флаг переноса.

Заметим, что RAL, ADC A и RAR сохраняют старое значение флага переноса (или в разряде 0; или в разряде 7), в то время как RLC, RRC и ADD A уничтожают его.

Примеры

1. Циклически сдвинуть аккумулятор вправо на две позиции без флага переноса:

```
RRC
RRC
```

2. Логически сдвинуть аккумулятор влево на две позиции:

```
ADD  A
ADD  A
```

Арифметический или логический сдвиг можно выполнить, используя флаг переноса. Команда STC устанавливает флаг переноса, в то время как ANA A или ORA A очищает его без изменения аккумулятора. Регистры или ячейки памяти могут быть сдвинуты с помощью пересылки их содержимого в аккумулятор. Единственный короткий путь выполнения 16-разрядного логического сдвига влево пары регистров H – это команда DAD H.

Примеры

1. Логически сдвинуть регистр C вправо на одну позицию:

```
MOV  A,C
ANA  A          ;ФЛАГ ПЕРЕНОСА = 0
RAR             ;СДВИНУТЬ ВПРАВО, ПЕРЕСЛАТЬ 0 В СТАРШИЙ БАЙТ
MOV  C,A
```

2. Сдвинуть ячейку памяти ADDR вправо на одну позицию, сохранив знаковый разряд (разряд 7). Перемещение, в результате которого сохраняется знаковый разряд, называется *расширением знака.* Сдвиг, который осуществляется подобным образом, называется *арифметическим*, так как он сохраняет

знак дополнительного кода числа. Поэтому его можно использовать для деления или нормализации чисел со знаками:

```
LDA  ADDR
RLC            ;СКОПИРОВАТЬ ЗНАКОВЫЙ РАЗРЯД ВО ФЛАГ ПЕРЕНОСА
               ; И РАЗРЯД 0
RAR            ;СДВИНУТЬ КОПИИ ВПРАВО ДВАЖДЫ
RAR
STA  ADDR
```

Эта последовательность команд построена на том, что RLC, как показано на рис. 1.3, пересылает разряд 7 как во флаг переноса, так и в разряд 0 аккумулятора. В результате получаются две копии разряда 7 – как раз то, что требуется для расширения знака.

1.9. ПРИНЯТИЕ РЕШЕНИЙ

Процедуры принятия решений могут быть классифицированы следующим образом:

переход, если разряд установлен (логическая единица) или очищен (логический нуль);

переход, если два значения равны или не равны;

переход, если одно значение больше другого или меньше его.

Наличие процедур первого класса позволяет процессору реагировать на значения флагов, переключателей, линии состояния или других двоичных (включено-выключено) сигналов. Наличие процедур второго класса позволяет процессору определить, имеет ли вводимая величина или результат определенное значение (например, введен ли определенный символ команды или терминатор, или равен ли результат нулю). Наличие процедур третьего класса позволяет процессору определить, превышает ли значение некоторый числовой порог или ниже его (например, правильное или ошибочное значение, выше или ниже предупредительного уровня или заданной точки). Если предположить, что основное значение находится в аккумуляторе, а вторичное значение (если оно необходимо) в каком-либо регистре или ячейке памяти, то эта процедура выглядит следующим образом.

1.9.1. ПЕРЕХОД, ЕСЛИ РАЗРЯД УСТАНОВЛЕН ИЛИ ОЧИЩЕН

Требуется определить, установлен или очищен какой-либо разряд, с помощью операции логическое И аккумулятора с числом, характеризуемым единицей в заданном разряде и нулями в остальных. Флаг нуля отражает значение разряда и может быть использован для условного перехода.

Примеры

1. Перейти к DEST, если разряд 5 аккумулятора равен 1:

```
ANI  00100000B      ;МАСКА ДЛЯ РАЗРЯДА 5
JNZ  DEST
```

Флаг нуля устанавливается в 1, если и только если разряд 5 аккумулятора равен 0.

2. Перейти к DEST, если разряд 2 аккумулятора равен 0:

```
ANI  00000100B        ;МАСКА ДЛЯ РАЗРЯДА 2
JZ   DEST
```

Для разрядов 0, 6 и 7 существует более короткий способ:

3. Перейти к DEST, если разряд 7 аккумулятора равен 1:

```
ANA  A                ;ОПРЕДЕЛИТЬ ФЛАГ ЗНАКА ПО A
JM   DEST
```

4. Перейти к DEST, если разряд 6 аккумулятора равен 0:

```
ADD  A                ;ОПРЕДЕЛИТЬ ФЛАГ ЗНАКА ПО РАЗРЯДУ 6
JP   DEST
```

5. Перейти к DEST, если разряд 0 аккумулятора равен 1:

```
RAR                   ;ПЕРЕСЛАТЬ РАЗРЯД 0 ВО ФЛАГ ПЕРЕНОСА
JC   DEST
```

1.9.2. ПЕРЕХОД, ОСНОВАННЫЙ НА РАВЕНСТВЕ

С помощью вычитания требуется определить, равно ли значение аккумулятора некоторой другой величине. При вычитании флаг нуля устанавливается в 1, если значения равны, и в 0, если они не равны. Команды сравнения (CMP или CPI) более удобны, чем команды вычитания (SBB, SBI, SUB или SUI), так как при сравнении аккумулятор сохраняется для последующих операций.

Примеры

1. Перейти к DEST, если аккумулятор содержит число VALUE:

```
CPI  VALUE            ;ДАННЫЕ = VALUE?
JZ   DEST             ;ДА, ПЕРЕЙТИ
```

2. Перейти к DEST, если содержимое аккумулятора не равно содержимому ячейки памяти ADDR:

```
LXI  H,ADDR
CMP  M                ;ДАННЫЕ = ЗНАЧЕНИЕ В ПАМЯТИ?
JNZ  DEST             ;НЕТ, ПЕРЕЙТИ
```

Если VALUE равно 0, 1, или FF_{16}, то есть более короткий путь.

3. Перейти к DEST, если аккумулятор содержит 0:

```
ANA  A                ;ОПРЕДЕЛИТЬ ФЛАГ ЗНАКА ПО A
JZ   DEST             ;ПЕРЕЙТИ, ЕСЛИ A СОДЕРЖИТ НУЛЬ
```

4. Перейти к DEST, если аккумулятор не содержит FF_{16}:

```
INR  A                ;ОПРЕДЕЛИТЬ ФЛАГ НУЛЯ
JNZ  DEST             ;ПЕРЕЙТИ, ЕСЛИ A НЕ БЫЛО РАВНО FF
```

Эта процедура применима к любому 8-разрядному регистру или ячейке памяти, адрес которой содержится в регистрах H и L.

5. Перейти к DEST, если аккумулятор содержит 1:

```
DCR  A                ;ОПРЕДЕЛИТЬ ФЛАГ НУЛЯ
JZ   DEST             ;ПЕРЕЙТИ, ЕСЛИ A СОДЕРЖИТ 1
```

6. Перейти к DEST, если ячейка памяти ADDR содержит 0:

```
LXI   H,ADDR
INR   M              ;ОПРЕДЕЛИТЬ ФЛАГ НУЛЯ ПО ADDR
DCR   M
JZ    DEST           ;ПЕРЕЙТИ, ЕСЛИ ADDR СОДЕРЖИТ НУЛЬ
```

Процедура "увеличить на 1, уменьшить на 1" применима также к любому регистру общего назначения.

1.9.3. ПЕРЕХОД, ОСНОВАННЫЙ НА СРАВНЕНИИ АБСОЛЮТНЫХ ВЕЛИЧИН

Определить с помощью вычитания, больше или меньше содержимое аккумулятора некоторой другой величины. Если, как это часто бывает, значения беззнаковые, флаг переноса указывает, какая из величин больше. Обычно:

флаг переноса =1, если вычитаемое значение больше, чем содержимое аккумулятора (т. е. если необходим заем разряда);

флаг переноса =0, если содержимое аккумулятора больше или обе величины равны (т. е. если нет необходимости в заеме разряда).

Так как вычитание равных значений очищает флаг переноса, возможны следующие варианты (при условии, что аккумулятор является первичным операндом);

первичный операнд меньше вторичного (флаг переноса установлен);

первичный операнд больше вторичного или равен ему (флаг переноса очищен).

Если требуются варианты "меньше чем или равны" и "больше чем" то необходимо просто поменять местами первичный и вторичный операнды (т. е. принять Y – X вместо X – Y). Другой подход состоит в определении случая равенства с помощью отдельного условного перехода.

Примеры

1. Перейти к DEST, если содержимое аккумулятора больше числа VALUE или равно ему:

```
CPI   VALUE          ;ДАННЫЕ БОЛЬШЕ VALUE?
JNC   DEST           ;ДА, ПЕРЕЙТИ
```

2. Перейти к DEST, если содержимое памяти с адресом OPER1 меньше содержимого памяти по адресу OPER2:

```
LDA   OPER1          ;ВЗЯТЬ ПЕРВЫЙ ОПЕРАНД
LXI   H,OPER2
CMP   M              ;ПЕРВЫЙ ОПЕРАНД МЕНЬШЕ ВТОРОГО?
JC    DEST           ;ДА, ПЕРЕЙТИ
```

Если значения содержат знак, то вычитание может вызвать переполнение по дополнительному коду, т. е. разность не будет помещаться в 7 разрядов и, следовательно, может изменить знаковый разряд. Однако в описанных далее случаях переполнения не произойдет, так что для условного перехода вместо флага переноса может быть использован флаг знака:

• Если два числа имеют одинаковый знак, то абсолютная величина разности меньше, чем наибольший (по абсолютной величине) операнд, и переполнения не может произойти. Имеют ли два числа одинаковый знак, легко опре-

делить с помощью операции ИСКЛЮЧАЮЩЕЕ ИЛИ между ними и проверки флага знака. Напомним, что операция ИСКЛЮЧАЮЩЕЕ ИЛИ двух разрядов дает 1, если и только если оба они имеют разное значение:

```
XRI  VALUE          ;МОЖЕТ ПРОИЗОЙТИ ПЕРЕПОЛНЕНИЕ?
JP   NOOVF          ;НЕ МОЖЕТ, ЕСЛИ ЗНАКИ ОДИНАКОВЫЕ
```

• Если значение, с которым вы сравниваете, равно 0, то установите и проверьте флаг знака.

Примеры

1. Перейти к DEST, если аккумулятор содержит положительное число со знаком:

```
ANA  A              ;УСТАНОВИТЬ ФЛАГИ ПО ЗНАЧЕНИЮ A
JP   DEST
```

2. Перейти к DEST, если регистр содержит отрицательное число со знаком:

```
INR  REG            ;УСТАНОВИТЬ ФЛАГИ ПО ЗНАЧЕНИЮ РЕГИСТРА
DCR  REG
JM   DEST
```

Эта последовательность команд не изменяет аккумулятор или регистр.

3. Перейти к DEST, если ячейка памяти ADDR содержит положительное число со знаком:

```
LXI  H,ADDR         ;ВЗЯТЬ АДРЕС ДАННЫХ В ПАМЯТИ
INR  M              ;УСТАНОВИТЬ ФЛАГИ ПО СОДЕРЖИМОМУ ПАМЯТИ
DCR  M
JP   DEST
```

Эта последовательность команд не изменяет аккумулятор или ячейку памяти.

Проверить, не произошло ли переполнения, можно, сравнив знаки результата и начального значения аккумулятора. Если из аккумулятора вычитается число с другим знаком и происходит переполнение, то, очевидно, что знак разности должен быть таким же, как и исходный знак аккумулятора. Для того чтобы проверить, являются ли два знака одинаковыми, воспользуйтесь последовательностью команд, вычитающей B из A и использующей C для временного запоминания:

```
MOV  C,A            ;СОХРАНИТЬ ИСХОДНОЕ ЗНАЧЕНИЕ В C
SUB  B              ;ВЫПОЛНИТЬ СРАВНЕНИЕ
XRA  C              ;ЗНАКИ ОТЛИЧАЮТСЯ?
JP   NOOVF          ;НЕТ, ПЕРЕПОЛНЕНИЯ НЕ БЫЛО
```

Эта последовательность команд неудобна, так как должны рассматриваться несколько случаев. Кроме того, исходные переменные и разности должны или сохраняться, или вычисляться заново. Процедура принятия решения при переполнении после сложения отличается от описанной выше, так как в этом случае переполнение может произойти только тогда, когда операнды имеют одинаковый знак.

В табл. 1.7 и 1.8 приводится краткая сводка обычных последовательностей команд принятия решения для микропроцессора 8080 или 8085. В табл. 1.7 даются последовательности, которые зависят только от значения аккуму-

Таблица 1.7. *Последовательности команд принятия решения, зависящие только от аккумулятора*

Условие	Команда, устанавливающая флаги	Условный переход
Любой из разрядов = 0	ANI маска (1 в позиции разряда)	JZ
Любой из разрядов = 1	ANI маска (1 в позиции разряда)	JNZ
Разряд 7 = 0	RAL, RLC или ADD A	JNC
Разряд 7 = 1	RAL, RLC или ADD A	JC
Разряд 6 = 0	ADD A	JP
Разряд 6 = 1	ADD A	JM
Разряд 0 = 0	RAR или RRC	JNC
Разряд 0 = 1	RAR или RRC	JC
(A) = 0	ANA A или ORA A	JZ
(A) ≠ 0	ANA A или ORA A	JNZ
(A) положительное (старший разряд = 0)	ANA A или ORA A	JP
(A) отрицательное (старший разряд = 1)	ANA A или ORA A	JM

Таблица 1.8. *Последовательности команд принятия решения, зависящие от результата сравнения чисел*

Условия	Команда, устанавливающая флаги	Условный переход
(A) = VALUE	CPI VALUE	JZ
(A) ≠ VALUE	CPI VALUE	JNZ
(A) ⩾ VALUE (без знака)	CPI VALUE	JNC
(A) < VALUE (без знака)	CPI VALUE	JC
(A) = (REG)	CMP REG	JZ
(A) ≠ (REG)	CMP REG	JNZ
(A) ⩾ (REG) (без знака)	CMP REG	JNC
(A) < (REG) (без знака)	CMP REG	JC

лятора; в табл. 1.8 приводятся последовательности команд, основанные на сравнении значений аккумулятора и заданного числа или какого-либо регистра. Если задан регистр M, то сравнение происходит с ячейкой памяти, адресуемой с помощью регистров H и L.

1.10. ОРГАНИЗАЦИЯ ЦИКЛОВ

Самый простой способ выполнения цикла (т. е. повторения последовательности команд) в микропроцессоре 8080 или 8085 состоит в следующем:

1) загрузить в регистр общего назначения число, указывающее, сколько раз должна быть выполнена последовательность команд;
2) выполнить команды;
3) уменьшить заданный регистр на 1;
4) вернуться к шагу 2, если результат шага 3 не равен 0.

Обычно подобные программы имеют следующий вид:

```
         MVI  REG,NTIMES      ;СЧЕТЧИК = ЧИСЛО ПОВТОРЕНИЙ
LOOP:    .
         .
         .    ПОВТОРЯЕМЫЕ КОМАНДЫ
         .
         .
         DCR  REG
         JNZ  LOOP
```

Команда уменьшения счетчика выбрана здесь только для наглядности. Можно увеличивать счетчик (используя команду INR); естественно, что в этом случае соответствующим образом должно быть изменено задание начального значения счетчика. В любом случае повторяемые команды не должны влиять на счетчик повторений. Счетчик может запоминаться в любом регистре общего назначения, однако большинство программистов используют для счетчика регистры B или C, так как H и L обычно служат в качестве первичного регистра адреса памяти, а D и E — в качестве дополнительного регистра адреса памяти.

Таким образом, типичные команды организации цикла выглядят следующим образом:

```
         MVI  B,NTIMES
LOOP:    .
         .
         .    ПОВТОРЯЕМЫЕ КОМАНДЫ
         .
         .
         DCR  B
         JNZ  LOOP
```

Длина регистра B, равная восьми разрядам, ограничивает этот простой цикл до 256 повторений. Как показано в следующих примерах, программист может организовать большее число повторений с помощью вложения циклов, в каждом из которых в качестве счетчика используется один регистр, или с помощью пары регистров.

• Вложенные циклы:

```
         MVI  B,NTIMM         ;НАЧАЛЬНОЕ ЗНАЧЕНИЕ ВНЕШНЕГО СЧЕТЧИКА
LOOPO:   MVI  C,NTIML         ;НАЧАЛЬНОЕ ЗНАЧЕНИЕ ВНУТРЕННЕГО СЧЕТЧИКА
LOOPI:   .
         .
         .    ПОВТОРЯЕМЫЕ КОМАНДЫ
         .
         DCR  C               ;УМЕНЬШИТЬ ВНУТРЕННИЙ СЧЕТЧИК
         JNZ  LOOPI
         DCR  B               ;УМЕНЬШИТЬ ВНЕШНИЙ СЧЕТЧИК
         JNZ  LOOPO
```

Во внешнем цикле после каждого уменьшения внешнего счетчика (регистр B) восстанавливается начальное значение внутреннего счетчика (NTIML) в регистре C. Результатом вложения циклов является умножение счетчиков — процессор повторяет команды, начинающиеся с LOOPI, NTIMM × NTIML раз.

• 16-разрядный счетчик в паре регистров:

```
          LXI  B,NTIMES       ;ИНИЦИАЛИЗИРОВАТЬ 16-РАЗРЯДНЫЙ СЧЕТЧИК
LOOP:     .
          .
          .    ПОВТОРЯЕМЫЕ КОМАНДЫ
          .
          DCX  B
          MOV  A,B            ;ПРОВЕРИТЬ 16-РАЗРЯДНЫЙ СЧЕТЧИК НА НУЛЬ
          ORA  C
          JNZ  LOOP
```

Эти дополнительные шаги необходимы вследствие того, что команда DCX не влияет на флаг нуля (т. е. никак нельзя сообщить, что счетчик стал равным нулю). Определить, что пара регистров содержит 0, проще всего с помощью команды логическое ИЛИ двух регистров. Результат равен 0 тогда и только тогда, когда все разряды обоих регистров содержат нули. Проверьте эту процедуру на работоспособность. Единственная проблема состоит в том, что в команде логическое ИЛИ используется аккумулятор, а это требует от Вас сохранять его старое содержимое, если оно необходимо для следующей итерации.

1.11. ОБРАБОТКА МАССИВОВ

Элемент массива проще всего выбрать, поместив его адрес в регистры H и L. В этом случае можно:

работать с элементом, обращаясь к нему как к регистру M;

выбирать следующий элемент массива (по следующему большему адресу); используя команду INX для увеличения значения пары регистров H и L, или предыдущий элемент (по предыдущему меньшему адресу), используя DCX для уменьшения H и L;

выбирать любой произвольный элемент, загрузив в другую пару регистров смещение элемента относительно адреса, содержащегося в HL, и использовав команду DAD (16-разрядное сложение).

Если массив одномерный и каждый элемент занимает один байт, то типичные процедуры работы с массивами легко программируются. Далее приводится несколько примеров.

• Добавить элемент массива к аккумулятору. Предполагается, что адрес этого элемента находится в регистрах H и L. Изменить H и L таким образом, чтобы они содержали адрес следующего 8-разрядного элемента:

```
          ADD  M         ;ДОБАВИТЬ ТЕКУЩИЙ ЭЛЕМЕНТ
          INX  H         ;УСТАНОВИТЬ АДРЕС СЛЕДУЮЩЕГО ЭЛЕМЕНТА
```

• Проверить элемент массива и, если он равен 0, добавить 1 к регистру B. Предполагается, что адрес элемента содержится в регистрах H и L. Изменить H и L таким образом, чтобы они содержали адрес предыдущего 8-разрядного элемента:

```
          MOV  A,M       ;ПОЛУЧИТЬ ТЕКУЩИЙ ЭЛЕМЕНТ
          ANA  A         ;ТЕКУЩИЙ ЭЛЕМЕНТ РАВЕН НУЛЮ?
          JNZ  UPDDT
          INR  B         ;ДА, ДОБАВИТЬ 1 К СЧЕТЧИКУ НУЛЕЙ
UPDDT:    DCX  H         ;УСТАНОВИТЬ АДРЕС ПРЕДЫДУЩЕГО ЭЛЕМЕНТА
```

• Загрузить в аккумулятор 35-й элемент массива. Предполагается, что H и L содержат базовый адрес массива:

```
LXI  D,35      ;ВЗЯТЬ СМЕЩЕНИЕ ДЛЯ 35-ГО ЭЛЕМЕНТА
DAD  D         ;ВЫЧИСЛИТЬ АДРЕС ЭЛЕМЕНТА
MOV  A,M       ;ПОЛУЧИТЬ ЭТОТ ЭЛЕМЕНТ
```

Команда DAD выполняет 16-разрядное сложение, используя H и L как 16-разрядный аккумулятор. 16-разрядное смещение в D и E может быть как положительным, так и отрицательным.

Работа с элементами массива усложняется, если во время каждой итерации необходим доступ более чем к одному элементу (как, например, в сортировке, которая требует перестановки элементов), длина элементов превышает один байт, или же сами элементы являются адресами (как в случае таблицы начальных адресов). Основная проблема состоит в отсутствии неявной индексации (которая могла бы позволить процессору вычислять в цикле команды индексированный адрес) и в отсутствии команд, содержащих косвенное обращение к 16-разрядным элементам. Здесь приводятся некоторые примеры более общей работы с массивами.

• Загрузить в D и E 16-разрядный элемент массива (младший байт запоминается первым). Предполагается, что H и L содержат начальный адрес элемента. Изменить H и L таким образом, чтобы они содержали начальный адрес следующего 16-разрядного элемента:

```
MOV  E,M       ;ВЗЯТЬ МЛАДШИЙ БАЙТ ЭЛЕМЕНТА
INX  H
MOV  D,M       ;ВЗЯТЬ СТАРШИЙ БАЙТ ЭЛЕМЕНТА
INX  H         ;УСТАНОВИТЬ АДРЕС СЛЕДУЮЩЕГО ЭЛЕМЕНТА
```

• Заменить элемент массива следующим элементом, если значения этих элементов не убывают. Предполагается, что элементы содержат 8-разрядные числа без знака и что адрес текущего элемента находится в H и L. Изменить H и L таким образом, чтобы они содержали адрес следующего элемента:

```
        MOV  A,M       ;ВЗЯТЬ ТЕКУЩИЙ ЭЛЕМЕНТ
        INX  H
        CMP  M         ;ЭТОТ ЭЛЕМЕНТ МЕНЬШЕ СЛЕДУЮЩЕГО?
        JNC  DONE      ;НЕТ, ПЕРЕСТАНОВКА НЕ НУЖНА
        MOV  B,M       ;ДА, НАЧАЛО ПЕРЕСТАНОВКИ
        MOV  M,A       ;ТЕКУЩИЙ ЭЛЕМЕНТ НА НОВОЕ МЕСТО
        DCX  H
        MOV  M,B       ;СЛЕДУЮЩИЙ ЭЛЕМЕНТ НА НОВОЕ МЕСТО
        INX  H
DONE:   NOP
```

Эта процедура неудобна, поскольку процессор, используя регистры H и L, может за раз адресовать только один элемент. Понятно, что эта проблема была бы еще более серьезной, если бы два элемента не были соседними.

• Загрузить аккумулятор из ячейки, косвенный адрес которой находится на 12-м месте в таблице. Предполагается, что базовый адрес таблицы находится в регистрах H и L:

```
LXI  D,24      ;ВЗЯТЬ УДВОЕННОЕ СМЕЩЕНИЕ ДЛЯ ЭЛЕМЕНТА
DAD  D         ;ВЫЧИСЛИТЬ НАЧАЛЬНЫЙ АДРЕС ЭЛЕМЕНТА
MOV  E,M       ;ВЗЯТЬ МЛАДШИЙ БАЙТ КОСВЕННОГО АДРЕСА
INX  H
```

```
MOV  D,M        ;ВЗЯТЬ СТАРШИЙ БАЙТ КОСВЕННОГО АДРЕСА
LDAX D          ;ПОЛУЧИТЬ ДАННЫЕ ИЗ КОСВЕННОГО АДРЕСА
```

Заметим, что при работе с таблицами, содержащими адреса, индекс должен удваиваться, так как каждый 16-разрядный адрес занимает два байта памяти.

Некоторые методы упрощения обработки массивов состоят в следующем.

- Базовый адрес таблицы следует держать в регистрах D и E (или B и C), поскольку команда DAD не изменяет их.
- Для удвоения индекса в аккумуляторе следует использовать команду ADD A. Этот удвоенный индекс можно затем использовать для работы с таблицами, содержащими 16-разрядные элементы.
- Для пересылки адресов в регистры H и L и из них следует использовать команду XCHG.

В гл. 5 и 9 приведены другие примеры работы с массивами.

1.12. ПОИСК В ТАБЛИЦЕ

Так как в процессорах 8080 и 8085 отсутствует индексация, то адрес, необходимый для поиска в таблице, должен вычисляться явно с использованием команды DAD. Как и при работе с массивами, поиск в таблице является простым, если таблица содержит 8-разрядные элементы данных; сложнее, когда таблица содержит более длинные элементы или же адреса. В этом случае могут быть полезны команды XCHG, PCHL и SPHL, но они требуют, чтобы программист помещал результаты в определенные пары регистров. Далее приводятся некоторые примеры.

- Загрузить элемент из таблицы в аккумулятор. Предполагается, что базовый адрес таблицы находится в BASE (постоянное значение), а 16-разрядный индекс – в ячейках памяти INDEX и INDEX + 1 (в INDEX + 1 старший байт):

```
LXI  D,BASE     ;ВЗЯТЬ КОСВЕННЫЙ АДРЕС
LHLD INDEX      ;ВЗЯТЬ ИНДЕКС
DAD  D          ;ВЫЧИСЛИТЬ АДРЕС ЭЛЕМЕНТА
MOV  A,M        ;ПОЛУЧИТЬ ЭТОТ ЭЛЕМЕНТ
```

Заметим, что изменить назначение пар регистров D и H не просто, так как отсутствуют команды, которые прямо загружали бы пару регистров D.

- Загрузить элемент из таблицы в аккумулятор. Предполагается, что базовый адрес таблицы находится в BASE (постоянное значение), а индекс – в аккумуляторе:

```
MOV  L,A        ;РАСШИРИТЬ ИНДЕКС ДО 16 РАЗРЯДОВ В HL
MVI  H,0
LXI  D,BASE     ;ВЗЯТЬ БАЗОВЫЙ АДРЕС
DAD  D          ;ВЫЧИСЛИТЬ АДРЕС ЭЛЕМЕНТА
MOV  A,M        ;ПОЛУЧИТЬ ЭТОТ ЭЛЕМЕНТ
```

- Загрузить 16-разрядный элемент из таблицы в регистры D и E. Предполагается, что базовый адрес таблицы находится в BASE (постоянное значение), а индекс – в аккумуляторе:

```
ADD  A          ;УДВОИТЬ ИНДЕКС ДЛЯ 16-РАЗРЯДНЫХ ЭЛЕМЕНТОВ
MOV  L,A        ;РАСШИРИТЬ ИНДЕКС ДО 16 РАЗРЯДОВ
MVI  H,0
LXI  B,BASE     ;ВЗЯТЬ БАЗОВЫЙ АДРЕС
```

```
        DAD  B          ;ВЫЧИСЛИТЬ НАЧАЛЬНЫЙ АДРЕС ЭЛЕМЕНТА
        MOV  E,M        ;ВЗЯТЬ МЛАДШИЙ БАЙТ ЭЛЕМЕНТА
        INX  H
        MOV  D,M        ;ВЗЯТЬ СТАРШИЙ БАЙТ ЭЛЕМЕНТА
```

Для удвоения индекса может быть использована также команда DAD H; в этом случае работа осуществляется медленнее, чем при ADD A, но автоматически выполняется обработка, когда удвоенный индекс слишком велик для восьми разрядов.

• Передать управление (перейти) на 16-разрядный адрес, полученный из таблицы. Предполагается, что базовый адрес таблицы находится в BASE (постоянное значение), а индекс – в аккумуляторе:

```
        ADD  A          ;УДВОИТЬ ИНДЕКС ДЛЯ 16-РАЗРЯДНЫХ ЭЛЕМЕНТОВ
        MOV  L,A        ;РАСШИРИТЬ ИНДЕКС ДО 16 РАЗРЯДОВ
        MVI  H,0
        LXI  B,BASE     ;ВЗЯТЬ БАЗОВЫЙ АДРЕС
        DAD  B          ;ВЫЧИСЛИТЬ НАЧАЛЬНЫЙ АДРЕС ЭЛЕМЕНТА
        MOV  A,M        ;ВЗЯТЬ МЛАДШИЙ БАЙТ АДРЕСА НАЗНАЧЕНИЯ
        INX  H
        MOV  H,M        ;ВЗЯТЬ СТАРШИЙ БАЙТ АДРЕСА НАЗНАЧЕНИЯ
        MOV  L,A
        PCHL            ;ПЕРЕЙТИ НА АДРЕС НАЗНАЧЕНИЯ
```

Таблицы переходов используются обычно для выполнения операторов CASE (многоходовые переходы, применяемые в таких языках, как Фортран, Паскаль и ПЛ/1) при декодировании команд, введенных с клавиатуры, и для ответа на функциональные клавиши терминала.

1.13. РАБОТА С СИМВОЛАМИ

Простейший способ работы с кодами символов в процессорах 8080 и 8085 состоит в обращении с ними как с 8-разрядными числами без знака. Буквы и цифры составляют упорядоченную последовательность набора символов в коде ASCII (например, представление буквы A в коде ASCII на единицу меньше, чем представление буквы B). Приложение В содержит полный набор символов ASCII.

Примеры

Перейти по адресу DEST, если аккумулятор содержит символ E в коде ASCII:

```
        CPI  'E'        ;ДАННЫЕ - СИМВОЛ E В КОДЕ ASCII?
        JZ   DEST       ;ДА, ПЕРЕЙТИ
```

Просмотреть строку, начиная с адреса STRNG для поиска символа, не являющегося пробелом:

```
        LXI  H,STRNG    ;УСТАНОВИТЬ АДРЕС НАЧАЛА СТРОКИ
EXAMC:  MOV  A,M        ;ВЗЯТЬ СЛЕДУЮЩИЙ СИМВОЛ
        CPI  ' '        ;ПРОБЕЛ?
        JNZ  DONE       ;НЕТ, СИМВОЛ НАЙДЕН
        INX  H          ;ДА, ПЕРЕЙТИ К СЛЕДУЮЩЕМУ СИМВОЛУ
        JMP  EXAMC
DONE:   NOP
```

или

```
         LXI   H,STRNG-1 ;УСТАНОВИТЬ АДРЕС БАЙТА,
                         ; ПРЕДШЕСТВУЮЩЕГО СТРОКЕ
EXAMC:   INX   H
         MOV   A,M       ;ВЗЯТЬ СЛЕДУЮЩИЙ СИМВОЛ
         CPI   ' '       ;ПРОБЕЛ?
         JZ    EXAMC     ;ДА, ПРОДОЛЖИТЬ ПОИСК
```

Любой из вариантов мог бы выполняться быстрее, если поместить пробел в какой-либо регистр общего назначения (например, в регистр C) и сравнивать каждый символ с этим регистром (используя команду CMP C), а не со значением непосредственно заданных данных.

Перейти по адресу DEST, если аккумулятор содержит букву между C и F включительно:

```
         CPI   'C'       ;СИМВОЛ МЕНЬШЕ C?
         JC    DONE      ;ДА
         CPI   'F'+1     ;СИМВОЛ РАВЕН ИЛИ МЕНЬШЕ F?
         JC    DEST      ;ДА, СИМВОЛ ДОЛЖЕН БЫТЬ МЕЖДУ C И F
DONE:    NOP
```

В гл. 8 приведены другие примеры работы со строками.

1.14. ПРЕОБРАЗОВАНИЕ КОДОВ

Данные могут быть преобразованы из одного кода в другой с помощью арифметических или логических операций (если соотношение кодов простое) или с помощью поиска в таблицах (если это соотношение сложное).

Примеры

1. Преобразовать цифру в коде ASCII в ее эквивалент в двоично-десятичном коде (BCD):

```
         SUI   '0'       ;ПРЕОБРАЗОВАТЬ ASCII В BCD
```

Так как цифры в ASCII составляют упорядоченную последовательность кодов, то необходимо всего лишь вычесть смещение (0 в коде ASCII).

Разряды 4 и 5 могут быть очищены с помощью команды:

```
         ANI   11001111B ;ПРЕОБРАЗОВАТЬ ASCII В BCD
```

Как арифметическая, так и логическая команды преобразуют, например, 0 ASCII (30_{16}) в десятичный 0 (00_{16}).

2. Преобразовать цифру, представленную в двоично-десятичном коде (BCD) в ее эквивалент в коде ASCII:

```
         ADI   '0'       ;ПРЕОБРАЗОВАТЬ BCD В ASCII
```

Обратное преобразование столь же простое. Вы можете установить также разряды 4 и 5 с помощью команды:

```
         ORI   00110000B ;ПРЕОБРАЗОВАТЬ BCD В ASCII
```

Как арифметическая, так и логическая команды преобразуют, например, 6 (06_{16}) в 6 ASCII (36_{16}).

3. Преобразовать 8-разрядный код в другой, используя поиск в таблице. Предполагается, что таблица начинается с адреса NEWCD и индексом являет-

ся значение исходного кода (например, 27-я запись, содержащая значение в новом коде, соответствует числу 27 в исходном коде). Кроме того, предполагается, что данные находятся в аккумуляторе:

```
        MOV  L,A        ;РАСШИРИТЬ ИНДЕКС ДО 16 РАЗРЯДОВ
        MVI  H,0
        LXI  D,NEWCD    ;ВЗЯТЬ БАЗОВЫЙ АДРЕС
        DAD  D          ;ВЫЧИСЛИТЬ АДРЕС ЭЛЕМЕНТА
        MOV  A,M        ;ВЗЯТЬ ЭЛЕМЕНТ
```

В гл. 4 содержатся другие примеры преобразования кодов.

1.15. АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ ПОВЫШЕННОЙ ТОЧНОСТИ

Арифметические операции повышенной точности требуют ряда 8-разрядных операций. Необходимо выполнить следующее:

сначала очистить флаг переноса, так как никогда не происходит переноса в младшие байты или заема из них;

использовать команды сложения с переносом (ADC) или вычитания с заемом (SBB) для выполнения 8-разрядных операций, которые включают перенос или заем из предыдущей операции.

Здесь приводится типовая программа 64-разрядного сложения:

```
        MVI  B,8        ;ЧИСЛО БАЙТОВ = 8
        ANA  A          ;ПЕРВОНАЧАЛЬНО ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА
        LXI  H,NUM1     ;УСТАНОВИТЬ АДРЕС НАЧАЛА ЧИСЕЛ
        LXI  D,NUM2
ADD8:   LDAX D          ;ВЗЯТЬ БАЙТ ОДНОГО ОПЕРАНДА
        ADC  M          ;ДОБАВИТЬ БАЙТ ДРУГОГО ОПЕРАНДА
        MOV  M,A        ;ПЕРЕСЛАТЬ 8-РАЗРЯДНУЮ СУММУ
        INX  D
        INX  H
        DCR  B          ;СОСЧИТАТЬ ЧИСЛО ОПЕРАЦИЙ С БАЙТАМИ
        JNZ  ADD8
```

В гл. 6 содержатся другие примеры.

1.16. УМНОЖЕНИЕ И ДЕЛЕНИЕ

Существует много способов выполнения умножения. Один из подходов состоит в замене умножения небольших целых чисел на определенную короткую последовательность сложений и сдвигов влево.

Примеры

1. Умножить аккумулятор на 2:

```
        ADD  A          ;УДВОИТЬ A
```

2. Умножить аккумулятор на 5:

```
        MOV  B,A
        ADD  A          ;УМНОЖИТЬ A НА 2
        ADD  A          ;УМНОЖИТЬ A НА 4
        ADD  B          ;УМНОЖИТЬ A НА 5
```

В обоих примерах предполагается, что переноса не происходит. Аналогично для получения 16-разрядного результата может быть использована команда DAD H.

Этот подход часто удобен для определения положения элемента в двумерных массивах. Например, допустим, что в каждом из трех различных резервуаров температура измерялась в четырех различных точках. Эти замеры организованы как двумерный массив T (I, J), где I — номер емкости (1, 2 или 3), а J указывает на определенную точку в резервуаре (1, 2, 3 или 4). Результаты замеров запоминаются в памяти с последовательными адресами, начиная с замера в точке 1 резервуара 1, следующим образом:

```
BASE      T(1,1)    ЗАМЕР В РЕЗЕРВУАРЕ 1, ТОЧКА 1
BASE+1    T(1,2)    ЗАМЕР В РЕЗЕРВУАРЕ 1, ТОЧКА 2
BASE+2    T(1,3)    ЗАМЕР В РЕЗЕРВУАРЕ 1, ТОЧКА 3
BASE+3    T(1,4)    ЗАМЕР В РЕЗЕРВУАРЕ 1, ТОЧКА 4
BASE+4    T(2,1)    ЗАМЕР В РЕЗЕРВУАРЕ 2, ТОЧКА 1
BASE+5    T(2,2)    ЗАМЕР В РЕЗЕРВУАРЕ 2, ТОЧКА 2
BASE+6    T(2,3)    ЗАМЕР В РЕЗЕРВУАРЕ 2, ТОЧКА 3
BASE+7    T(2,4)    ЗАМЕР В РЕЗЕРВУАРЕ 2, ТОЧКА 4
BASE+8    T(3,1)    ЗАМЕР В РЕЗЕРВУАРЕ 3, ТОЧКА 1
BASE+9    T(3,2)    ЗАМЕР В РЕЗЕРВУАРЕ 3, ТОЧКА 2
BASE+10   T(3,3)    ЗАМЕР В РЕЗЕРВУАРЕ 3, ТОЧКА 3
BASE+11   T(3,4)    ЗАМЕР В РЕЗЕРВУАРЕ 3, ТОЧКА 4
```

В общем случае результат замера T (I, J) располагается по адресу BASE + + 4 × (I − 1) + (J − 1). Если I находится в аккумуляторе, а J в регистре B, то T (I, J) можно загрузить в аккумулятор следующим образом:

```
DCR  A          ;ВЫЧИСЛИТЬ СМЕЩЕНИЕ ДЛЯ РЕЗЕРВУАРА I
ADD  A          ;2 X (I-1)
ADD  A          ;4 X (I-1)
ADD  B          ;ДОБАВИТЬ СМЕЩЕНИЕ ДЛЯ ТОЧКИ J
DCR  A          ;4 X (I-1) + (J-1)
MOV  L,A        ;РАСШИРИТЬ ИНДЕКС ДО 16 РАЗРЯДОВ
MVI  H,0
LXI  D,BASE     ;ВЗЯТЬ БАЗОВЫЙ АДРЕС РЕЗУЛЬТАТОВ ЗАМЕРОВ
DAD  D          ;ПОЛУЧИТЬ АДРЕС РЕЗУЛЬТАТА НУЖНОГО ЗАМЕРА
MOV  A,M        ;ВЫБРАТЬ НУЖНЫЙ РЕЗУЛЬТАТ
```

Этот подход может быть расширен, как показано в гл. 5, для многоразмерных массивов.

Деление на число, являющееся степенью 2, можно выполнить с помощью ряда логических сдвигов вправо.

Пример

Разделить аккумулятор на 4:

```
ANA  A          ;ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА
RAL             ;РАЗДЕЛИТЬ A НА 2
ANA  A          ;ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА
RAL             ;ЕЩЕ РАЗ РАЗДЕЛИТЬ A НА 2
```

или

```
RAL             ;ДВАЖДЫ СДВИНУТЬ ЦИКЛИЧЕСКИ ВПРАВО
RAL
ANI  00111111B  ;И ОЧИСТИТЬ ДВА НАЧАЛЬНЫХ РАЗРЯДА
```

Если умножаются или делятся числа со знаком, то необходимо обращать внимание на отделение знака от абсолютной величины. Логические сдвиги должны быть заменены на арифметические, сохраняющие знаковый разряд.

К другим методам умножения и деления относятся описанные в гл. 6 алгоритмы, связанные со сдвигами и сложениями (при умножении) или со сдвигами и вычитаниями (при делении), а также рассмотренный ранее в этой главе метод поиска в таблице.

1.17. ОБРАБОТКА СПИСКОВ

Дополнительная информация по приведенному далее материалу может быть найдена в статье [4].

Если элементы списков хранятся в памяти в последовательных адресах, то такие списки можно обрабатывать так же, как массивы. Если два массива образуют очереди или цепочки, то становится очевидной ограниченность данного набора команд, выражающаяся в следующем:

не предусмотрена индексация:

косвенная адресация возможна только через пары регистров;

адреса в парах регистров могут быть использованы только для получения или записи 8-разрядных данных.

Примеры

1. Получить адрес, который записан по адресу, находящемуся в регистрах H и L. Поместить полученный адрес в регистры H и L:

```
MOV  E,M       ;ВЗЯТЬ МЛАДШИЙ БАЙТ СВЯЗИ
INX  H
MOV  D,M       ;ВЗЯТЬ СТАРШИЙ БАЙТ СВЯЗИ
XCHG           ;ТЕКУЩИЙ УКАЗАТЕЛЬ ЗАМЕНИТЬ НА СВЯЗЬ
```

С помощью этой процедуры можно продвигаться в связанном списке от одного элемента к другому.

2. Получить данные, адрес которых хранится в ячейках памяти INDIR и INDIR + 1 (старший байт в INDIR + 1), и увеличить этот адрес на 1:

```
LHLD INDIR     ;ВЗЯТЬ УКАЗАТЕЛЬ ИЗ ПАМЯТИ
MOV  A,M       ;ВЗЯТЬ ДАННЫЕ, ИСПОЛЬЗУЯ УКАЗАТЕЛЬ
INX  H         ;УВЕЛИЧИТЬ УКАЗАТЕЛЬ НА 1
SHLD INDIR
```

Эта процедура позволяет использовать адрес, хранящийся в памяти, в качестве указателя для следующей ячейки памяти.

3. Адрес, содержащийся в DE, запомнить по адресу, который находится в регистрах H и L. Увеличить H и L на 2:

```
MOV  M,E       ;ЗАПОМНИТЬ МЛАДШИЙ БАЙТ УКАЗАТЕЛЯ
INX  H
MOV  M,D       ;ЗАПОМНИТЬ СТАРШИЙ БАЙТ УКАЗАТЕЛЯ
INX  H         ;ЗАКОНЧИТЬ ОБНОВЛЕНИЕ H И L
```

Эта процедура позволяет сформировать список адресов. Такой список можно было бы использовать, например, для записи связанного программного сегмента, в котором каждая программа по завершению выполнения передает управление следующей в списке. Такой список мог бы также содержать начальные адреса ряда тестовых процедур или задач или адреса памяти или устройства ввода-вывода, которым оператор присвоил определенные функции.

Дополнительная информация по приведенному ниже материалу может быть найдена в книге [5].

Более распространенные структуры данных можно обрабатывать с помощью процедур, предназначенных для работы с массивами, таблицами и списками. Основные недостатки набора команд в этом случае те же самые, что упоминались при рассмотрении обработки списков.

Примеры

1. **Очереди или связанные списки.** Допустим, что есть заголовок очереди, который содержит в ячейках памяти HEAD и HEAD + 1 базовый адрес первого элемента очереди. Если очередь пустая, то HEAD и HEAD + 1 содержат нули. Первые две ячейки каждого элемента содержат базовый адрес следующего элемента или 0, если следующий элемет отсутствует.

• Добавить элемент в начало очереди. Предполагается, что базовый адрес элемента находится в регистрах D и E:

```
        LXI   H,HEAD         ;ЗАМЕНИТЬ НАЧАЛО, ЗАПОМНИВ СТАРОЕ
                             ; ЗНАЧЕНИЕ
        MOV   A,M            ;ПЕРЕСЛАТЬ МЛАДШИЙ БАЙТ
        MOV   M,E
        INX   H
        MOV   B,M            ;ПЕРЕСЛАТЬ СТАРШИЙ БАЙТ
        MOV   M,D
        STAX  D              ;НОВОЕ НАЧАЛО УКАЗЫВАЕТ НА СТАРОЕ,
        MOV   A,B            ; ВКЛЮЧАЯ СТАРШИЕ БАЙТЫ
        INX   D
        STAX  D
```

• Если очередь пустая, установить флаг нуля и осуществить выход из программы. Иначе удалить элемент из начала очереди, поместить базовый адрес в регистры D и E и очистить флаг нуля:

```
        LXI   H,HEAD         ;ПОЛУЧИТЬ НАЧАЛО ОЧЕРЕДИ
        MOV   E,M            ;МЛАДШИЙ БАЙТ
        INX   H
        MOV   D,M            ;СТАРШИЙ БАЙТ
        MOV   A,D
        ORA   E              ;ЕСТЬ ЭЛЕМЕНТЫ В ОЧЕРЕДИ?
        JZ    DONE           ;НЕТ
        INX   D              ;ДА, СДЕЛАТЬ СЛЕДУЩЩИЙ ЭЛЕМЕНТ НОВЫМ
        LDAX  D              ; НАЧАЛОМ
        MOV   M,A            ;СТАРШИЙ БАЙТ
        DCX   D
        DCX   H
        LDAX  D
        MOV   M,A            ;МЛАДШИЙ БАЙТ
DONE:   NOP
```

Так как команды, следующие после ORA E, не изменяют флаги, значение флага нуля на выходе указывает на то, была ли очередь пустая.

2. **Стеки.** Допустим, что есть программный стек, содержащий 8-разрядные элементы. Адрес следующей пустой ячейки находится в ячейках с адресами SPTR и SPTR + 1. Наименьший адрес памяти, которую может занимать стек – LOW, а наибольший – HIGH. Заметим, что этот программный стек растет

вверх (в сторону больших адресов), в то время как аппаратный стек микропроцессора растет вниз (в сторону меньших адресов).

• Если стек переполняется, то установить флаг переноса и выйти из программы. Иначе запомнить аккумулятор в стеке и увеличить указатель стека на 1. Переполнение означает, что стек вышел за пределы отведенной для него области:

```
          LHLD SPTR          ;ВЗЯТЬ УКАЗАТЕЛЬ СТЕКА
          XCHG
          LXI  H,-(HIGH+1)   ;ПРОВЕРИТЬ СТЕК НА ПЕРЕПОЛНЕНИЕ
          DAD  D             ;УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА, ЕСЛИ ПЕРЕПОЛНЕН
          JC   DONE
          XCHG
          MOV  M,A           ;ЗАПОМНИТЬ АККУМУЛЯТОР В СТЕКЕ
          INX  H             ;ОБНОВИТЬ УКАЗАТЕЛЬ СТЕКА
          SHLD SPTR
DONE:     NOP
```

• Если стек опускается за пределы нижней границы, то установить флаг переноса и выйти из программы. Иначе уменьшить указатель стека на 1 и загрузить аккумулятор из стека. Выход за нижнюю границу означает, что Вы попытались удалить данные из пустого стека:

```
          LHLD SPTR          ;ВЗЯТЬ УКАЗАТЕЛЬ СТЕКА
          XCHG
          LXI  H,-(LOW+1)    ;ПРОВЕРИТЬ НА ВЫХОД ЗА НИЖНЮЮ ГРАНИЦУ
          DAD  D             ;ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА, ЕСЛИ ЕСТЬ ВЫХОД
          JNC  DONE
          XCHG
          DCX  H             ;ОБНОВИТЬ УКАЗАТЕЛЬ СТЕКА
          MOV  A,M           ;ЗАГРУЗИТЬ АККУМУЛЯТОР ИЗ СТЕКА
          SHLD SPTR          ;ВОССТАНОВИТЬ УКАЗАТЕЛЬ СТЕКА
DONE:     CMC                ;УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА ПРИ ВЫХОДЕ
                             ; ЗА НИЖНЮЮ ГРАНИЦУ
```

В обоих примерах программ использован тот факт, что команда DAD влияет только на флаг переноса. Заметим также, что DCX и INX не изменяют ни один флаг.

1.19. СПОСОБЫ ПЕРЕДАЧИ ПАРАМЕТРОВ

Наиболее общими способами передачи параметров в микропроцессорах 8080 или 8085 являются следующие:

1. В регистрах. Доступными являются семь 8-разрядных регистров общего назначения (A, B, C, D, E, H и L), при этом три пары регистров (B, D и H) могут служить для передачи адресов. Этот подход пригоден в простейших случаях, но он лишен общности и может использоваться только при ограниченном числе параметров. Программисту следует помнить стандартные случаи использования регистров при задании параметров. Иначе говоря:

аккумулятор является очевидным местом для помещения одного 8-разрядного параметра;

пара регистров H является очевидным местом для помещения основного параметра адресной длины (16 разрядов);

пара регистров D из-за возможности использования команды XCHG явля-

ется лучшим, чем пара регистров B, местом для помещения дополнительного параметра адресной длины.

Этот подход допускает повторное использование программного модуля, поскольку программы обслуживания прерываний сохраняют и восстанавливают все регистры.

2. В заданной области памяти. Для реализации этого подхода проще всего поместить базовый адрес заданной области в регистры H и L. Вызывающая программа до передачи управления подпрограмме должна хранить параметры в памяти и загрузить базовый адрес в регистры H и L. Этот подход является общим и пригоден при любом числе параметров, но требует большего умения. Если при каждом обращении задаются разные области памяти, то создается стек таких областей. Если используется общая область памяти, то теряется возможность повторного входа в подпрограмму. При использовании этого метода программист отвечает за присваивание областей памяти, исключение взаимного влияния программ и сохранение и восстановление указателей, необходимых для возобновления работы программ после обращений к подпрограммам или после прерываний.

3. В памяти программы непосредственно за вызовом подпрограммы. Если вы используете этот подход, то должны помнить следующее:

• Базовый адрес этой памяти находится в вершине стека. Таким образом, базовый адрес является обычным адресом возврата, т. е. адресом ячейки, расположенной сразу за вызовом подпрограммы. Базовый адрес можно переслать в регистры H и L, извлекая его из стека с помощью команды:

```
POP  H          ;ПОЛУЧИТЬ БАЗОВЫЙ АДРЕС ОБЛАСТИ ПАРАМЕТРОВ
```

• Все параметры для данного вызова должны быть неизменяемыми, так как программы обычно располагаются в постоянной памяти, из которой можно только читать.

• Перед выполнением команды RETURN подпрограмма должна вычислить действительный адрес возврата (адрес ячейки, следующей сразу за областью параметров) и поместить его в вершину стека.

Пример

Допустим, что подпрограмма SUBR требует 8- и 16-разрядный параметры. Покажем, как выглядит основная программа, которая вызывает SUBR и содержит требуемые параметры. Кроме того, покажем начальную часть подпрограммы, которая извлекает параметры, запоминает 8-разрядный элемент в аккумуляторе, а 16-разрядный – в регистрах H и L и помещает правильный адрес возврата в вершину стека.

Вызов подпрограммы:

```
CALL SUBR       ;ВЫПОЛНИТЬ ПОДПРОГРАММУ
DB   PAR8       ;8-РАЗРЯДНЫЙ ПАРАМЕТР
DW   PAR16      ;16-РАЗРЯДНЫЙ ПАРАМЕТР
...СЛЕДУЮЩИЕ КОМАНДЫ...
```

Подпрограмма:

```
SUBR:   POP  H          ;УСТАНОВИТЬ АДРЕС НАЧАЛА ОБЛАСТИ ПАРАМЕТРОВ
        MOV  A,M        ;ВЗЯТЬ 8-РАЗРЯДНЫЙ ПАРАМЕТР
        INX  H
```

```
MOV  E,M          ;ВЗЯТЬ МЛАДШИЙ БАЙТ 16-РАЗРЯДНОГО ПАРАМЕТРА
INX  H
MOV  D,M          ;ВЗЯТЬ СТАРШИЙ БАЙТ 16-РАЗРЯДНОГО ПАРАМЕТРА
INX  H            ;УСТАНОВИТЬ АДРЕС СЛЕДУЮЩЕЙ КОМАНДЫ
PUSH H            ;ВОССТАНОВИТЬ ПРАВИЛЬНЫЙ АДРЕС ВОЗВРАТА
XCHG              ;ПЕРЕСЛАТЬ 16-РАЗРЯДНЫЙ ПАРАМЕТР В HL
.
.
...ОСТАЛЬНАЯ ЧАСТЬ ПОДПРОГРАММЫ...
.
RET               ;ВЕРНУТЬСЯ В ВЫЗЫВАЮЩУЮ ПРОГРАММУ
```

Начальная команда POP H загружает в регистры H и L адрес возврата, который сохранила в вершине стека команда CALL SUBR. В действительности, по адресу возврата находится не команда, а первый параметр (PAR8). После трех команд INX H в H и L содержится адрес следующей выполняемой команды вызывающей программы. Команда PUSH H помещает этот адрес в вершину стека с тем, чтобы заключительная команда RET вернула управление команде, следующей за параметрами.

Этот подход позволяет иметь список параметров любой длины. Однако получение параметров из памяти и организация адреса возврата в лучшем случае неудобны, так как при увеличении числа параметров этот процесс удлиняется и замедляется.

4. В стеке. Пользуясь этим подходом, вы должны помнить следующее.

• При выполнении команды CALL запоминается адрес возврата в вершине стека. Параметры, которые помещает в стек передающая программа, должны начинаться с адреса ssss + 2, где ssss – содержимое указателя стека. 16-разрядный адрес занимает две ячейки в вершине стека, а сам указатель стека всегда указывает на самый низший (последний), а не на самый верхний (следующий) адрес.

• Значение указателя стека (т. е. положение параметров) можно определить в подпрограмме только с помощью последовательности команд

```
LXI  H,0        ;ПЕРЕСЛАТЬ УКАЗАТЕЛЬ СТЕКА В HL
DAD  SP
```

Эти команды помещают указатель стека в HL (в противоположность SPHL).

• Вызывающая программа должна поместить параметры в стек до вызова подпрограммы. Подпрограмма должна извлечь параметры из стека и поместить туда результаты до возврата управления. Переключение на подпрограмму большей части накладных расходов (относящихся часто к очистке стека) снижает накладные расходы, связанные с каждым вызовом подпрограммы.

• Полная возможность повторного использования подпрограммы может быть достигнута с помощью динамического распределения ячеек памяти, отведенных под стек, для временного хранения данных с помощью команд

```
LXI  H,-NTEMP   ;ОСТАВИТЬ МЕСТО ДЛЯ ВРЕМЕННЫХ ДАННЫХ
DAD  SP
SPHL            ;ОБЫЧНЫЙ СПОСОБ КОРРЕКТИРОВКИ SP
```

Как показано на рис. 1.6, эта последовательность команд оставляет в верхней части стека NTEMP свободных ячеек памяти. Конечно, если значение

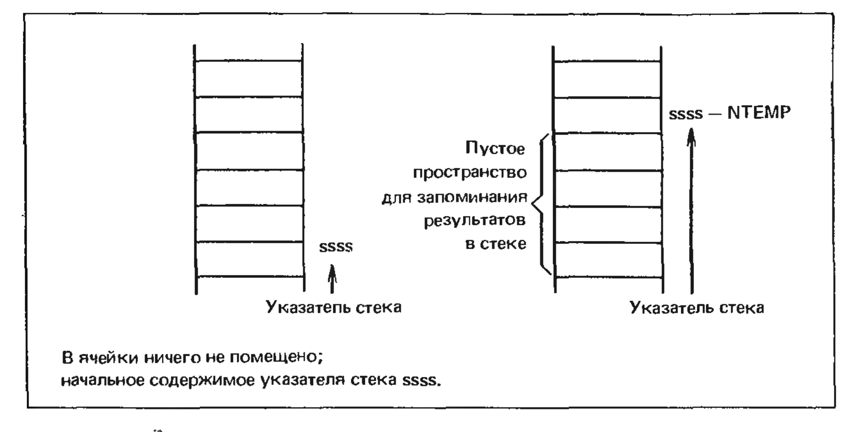


Рис. 1.6. Стек до и после выделения NTEMP пустых ячеек для временного хранения

Начальное состояние стека

ssss

Указатель стека

Стек после выполнения CALL SUBR

Младший байт адреса возврата

Старший байт адреса возврата

Пусто

8-разрядный параметр

Младший байт 16-разрядного параметра

Старший байт 16-разрядного параметра

ssss—6

Указатель стека

Конечное состояние стека

Результат 1

Результат 2

ssss—2

Указатель стека

Начальное содержимое указателя стека ssss

Рис. 1.7. Влияние подпрограммы на стек

NTEMP мало, то проще NTEMP раз выполнить команду DCX SP, что будет и быстрее и короче.

Пример

Допустим, что подпрограмме SUBR требуется два параметра: 8- и 16-разрядный, и что результатом ее выполнения являются два 8-разрядных числа. Показать вызов SUBR с передачей параметров через аккумулятор и регистры H и L и записью результатов перед вызовом из ячеек памяти RESLT и RESLT + 1 в стек. На рис. 1.7 показан вид стека в начале, после вызова подпрограммы и после возвращения из подпрограммы. Постоянно используя стек для параметров и результатов, Вы должны держать параметры в верхней части стека в соответствующем порядке.

Вызывающая программа:

```
        LHLD PAR16      ;ВЗЯТЬ 16-РАЗРЯДНЫЙ ПАРАМЕТР
        PUSH H          ;ПЕРЕСЛАТЬ 16-РАЗРЯДНЫЙ ПАРАМЕТР В СТЕК
        LDA  PAR8       ;ВЗЯТЬ 8-РАЗРЯДНЫЙ ПАРАМЕТР
        PUSH PSW        ;8-РАЗРЯДНЫЙ ПАРАМЕТР ЗАПИСАТЬ В СТЕК,
                        ; ТЕРЯЯ БАЙТ
        CALL SUBR       ;ВЫПОЛНИТЬ ПОДПРОГРАММУ
        NOP             ;ТЕПЕРЬ РЕЗУЛЬТАТЫ НАХОДЯТСЯ В ВЕРХНЕЙ
                        ; ЧАСТИ СТЕКА
```

Подпрограмма:

```
SUBR:   POP  B          ;СОХРАНИТЬ АДРЕС ВОЗВРАТА В ПАРЕ РЕГИСТРОВ BC
        POP  PSW        ;ВЗЯТЬ 8-РАЗРЯДНЫЙ ПАРАМЕТР
        POP  H          ;ВЗЯТЬ 16-РАЗРЯДНЫЙ ПАРАМЕТР
        PUSH B          ;ПОМЕСТИТЬ АДРЕС ВОЗВРАТА ОБРАТНО В СТЕК
        .
        ...ОСТАЛЬНАЯ ЧАСТЬ ПОДПРОГРАММЫ...
        .
        .
        LHLD RESLT      ;ВЗЯТЬ РЕЗУЛЬТАТ
        XTHL            ;РЕЗУЛЬТАТ В СТЕК, АДРЕС ВОЗВРАТА В HL
        PCHL            ;ВОЗВРАТИТЬСЯ В ОСНОВНУЮ ПРОГРАММУ
```

Команда XTHL помещает результат в стек и загружает адрес возврата в HL. Заметим, что можно передать в стек или из стека только 16-разрядную пару регистров; для работы с 8-разрядными параметрами проще всего или объединять их по два, или просто каждый раз терять в стеке один байт.

1.20. ПРОСТОЙ ВВОД-ВЫВОД

Простой ввод-вывод можно выполнить, используя или 8-разрядные адреса устройств, или 16-разрядные адреса памяти. Преимущества адресов устройств состоят в том, что они короче и обеспечивают отдельное адресное пространство для портов ввода-вывода. Недостаток состоит только в том, что команды IN и OUT содержат адреса устройств и допускают только прямую адресацию, т. е. команды IN и OUT требуют, чтобы были определены адреса; в этом случае отсутствует простой способ передачи адресов устройств ввода-вывода в виде параметров, так чтобы одна программа ввода-вывода поддерживала много устройств. С другой стороны, если порты ввода-вывода занимают адреса памяти, то с помощью любых команд, обращающихся к памяти, может

выполняться также ввод-вывод. Проблемы, связанные с этим подходом, состоят в его нестандартности, что создает трудности в тех случаях, когда надо отличить передачи ввода-вывода от передач в памяти, а также когда требуется, чтобы некоторая область памяти была зарезервирована для устройств ввода-вывода.

Примеры

1. Загрузить аккумулятор из порта ввода 2

```
        IN   2          ;ПРОЧИТАТЬ ИЗ ПОРТА 2
```

Адрес устройства (02_{16}) является частью памяти программы. Следовательно, его нельзя легко изменить для работы с другим набором портов ввода-вывода или с непостоянными устройствами ввода-вывода.

2. Загрузить аккумулятор из порта ввода, присвоенного адресу памяти, содержащемуся в регистрах H и L:

```
        MOV  A,M        ;ПРОЧИТАТЬ ДАННЫЕ ИЗ ПОРТА ВВОДА
```

Здесь одна и та же программа может получить данные из памяти с любым адресом. Конечно, теперь эти адреса памяти не доступны для нормального использования, что уменьшает действительный объем памяти ЭВМ.

3. Записать содержимое аккумулятора в порт вывода 6:

```
        OUT  6          ;ЗАПИСАТЬ ДАННЫЕ В ПОРТ 6
```

Константа 6 является частью памяти программы, и ее нельзя легко изменить для работы с другим набором портов или с непостоянными устройствами ввода-вывода.

4. Записать содержимое аккумулятора в порт вывода, присвоенный адресу памяти в регистрах H и L:

```
        MOV  M,A        ;ПОСЛАТЬ ДАННЫЕ В ПОРТ ВЫВОДА
```

Здесь одна и та же программа вывода может посылать данные в память с любым адресом. Действительный адрес может быть параметром, что позволяет программе работать с многочисленными устройствами ввода-вывода, присоединенными к одной ЭВМ, или с изменяющимися адресами в соответствии с различными конфигурациями или моделями.

5. Загрузить аккумулятор из памяти с адресом, полученным из таблицы устройств. Эта таблица начинается с адреса IOTBL, а номер устройства находится в аккумуляторе:

```
        ADD  A          ;УДВОИТЬ ИНДЕКС ДЛЯ 16-РАЗРЯДНЫХ ЭЛЕМЕНТОВ
        MOV  L,A        ;СДЕЛАТЬ ИНДЕКС В HL 16-РАЗРЯДНЫМ
        MVI  H,0
        LXI  D,IOTBL    ;ВЗЯТЬ БАЗОВЫЙ АДРЕС ТАБЛИЦЫ
        DAD  D          ;ПОЛУЧИТЬ АДРЕС АДРЕСА УСТРОЙСТВА
        MOV  E,M        ;ВЗЯТЬ МЛАДШИЙ БАЙТ АДРЕСА УСТРОЙСТВА
        INX  H
        MOV  D,M        ;ВЗЯТЬ СТАРШИЙ БАЙТ АДРЕСА УСТРОЙСТВА
        LDAX D          ;ПРОЧИТАТЬ ДАННЫЕ С УСТРОЙСТВА ВВОДА
```

Здесь опять предполагается, что устройство адресуется так же, как и ячейка памяти.

Таблица устройств ввода-вывода позволяет нам различать действительные

адреса ввода-вывода (*физические устройства*) и номера устройств, к которым обращается программа (*логические устройства*). Системная программа использует эту таблицу устройств ввода-вывода для превращения номеров устройств в действительные адреса ввода-вывода. Оператор или программист может в этом случае изменять назначение без знания процесса преобразования или аппаратуры ввода-вывода, скрывающейся за этим.

Например, программа, написанная на языке высокого уровня, может обращаться к устройству ввода номер 2 и устройству вывода номер 5. С целью проверки оператор может присвоить номера устройств 2 и 5 соответственно портам ввода и вывода своей консоли. Для нормальной автономной работы оператор может присвоить номер 2 устройству аналогового ввода, а номер 5 — системному устройству печати. Для работы с удаленной консоли оператор может присвоить номера 2 и 5 устройствам связи, используемым для ввода и вывода. В реальном применении таблица устройств обычно содержит начальные адреса подпрограмм ввода-вывода (*драйверов*), а не действительные адреса памяти.

В гл. 10 приведены дополнительные примеры программ ввода-вывода.

1.21. СОСТОЯНИЕ И УПРАВЛЕНИЕ

Сигналы состояния и управления могут обрабатываться так же, как любые другие данные. Единственная особенность состоит в том, что процессор не может читать из порта вывода; если необходимо знать текущее состояние порта вывода, Вы должны хранить копию данных в оперативной памяти.

Примеры

1. Перейти по адресу DEST, если разряд 3 порта ввода 6 равен 1:

```
IN    6              ;ПРОЧИТАТЬ СОСТОЯНИЕ ИЗ ПОРТА 6
ANI   00001000B      ;ПРОВЕРИТЬ РАЗРЯД 3
JNZ   DEST
```

2. Перейти по адресу DEST, если разряды 4, 5 и 6 порта ввода STAT равны 5 (двоичное число 101):

```
IN    STAT           ;ПРОЧИТАТЬ СОСТОЯНИЕ
ANI   01110000B      ;ВЫДЕЛИТЬ РАЗРЯДЫ 4, 5 И 6
CPI   01010000B      ;ПОЛЕ СОСТОЯНИЯ = 5?
JZ    DEST           ;ДА, ПЕРЕЙТИ НА DEST
```

3. Установить разряд 5 порта вывода CNTL в 1. Предполагается, что копия данных содержится в таблице, начинающейся по адресу OUTP:

```
LXI   H,OUTP+CNTL    ;ВЗЯТЬ КОПИЮ ДАННЫХ
MOV   A,M
ORI   00100000B      ;УСТАНОВИТЬ РАЗРЯД 5 ПОРТА
OUT   CNTL           ;ПОСЛАТЬ ДАННЫЕ В ПОРТ ВЫВОДА
MOV   M,A            ;ОБНОВИТЬ КОПИЮ ДАННЫХ
```

Копию необходимо обновлять каждый раз при обновлении данных.

4. Установить разряды 2, 3 и 4 порта вывода CNTL равными 6 (двоичное число 110). Предполагается, что копия данных содержится в таблице, начинающейся по адресу OUTP:

```
LXI   H,OUTP+CNTL    ;ВЗЯТЬ КОПИЮ ДАННЫХ
MOV   A,M
```

```
ANI  11100011B      ;ОЧИСТИТЬ РАЗРЯДЫ 2, 3 И 4
ORI  00011000B      ;УСТАНОВИТЬ ПОЛЕ УПРАВЛЕНИЯ РАВНЫМ 6
OUT  CNTL           ;ПОСЛАТЬ ДАННЫЕ В ПОРТ ВЫВОДА
MOV  M,A            ;ОБНОВИТЬ КОПИЮ ДАННЫХ
```

Сохранение копии данных в памяти (или использование значений, записанных в закрытый буферированный порт вывода), позволяет изменять часть данных без изменения остальной части, которая может иметь несвязанное с первой значение. Состояние одного светового индикатора (например, лампочки, которая индицирует локальную или дистанционную операцию) может быть изменено без воздействия на другие световые индикаторы, соединенные с тем же самым портом. Аналогично, сигнал в одной управляющей линии (например, в линии, которая определяет, в каком направлении по оси X двигался объект — положительном или отрицательном) можно было бы изменить без воздействия на другие управляющие линии, соединенные с тем же самым портом.

1.22. ПЕРИФЕРИЙНЫЕ ИНТЕГРАЛЬНЫЕ МИКРОСХЕМЫ

В системах 8080 и 8085 наиболее общими периферийными интегральными микросхемами являются последовательный интерфейс 8251, программируемый таймер 8253 и параллельный интерфейс 8255. Все эти устройства могут выполнять множество функций, большинство из которых подобно функциям самого микропроцессора. Конечно, периферийные интегральные микросхемы выполняют меньше различных функций, чем процессор, и диапазон этих функций существенно уже.

Идея программируемых интегральных микросхем состоит в том, что каждая микросхема содержит много схем; разработчик выбирает те из них, которые считает нужным использовать, с помощью записи необходимых кодов в управляющие регистры почти так же, как выбирают схемы из справочника проектировщика с помощью задания номеров страниц или других обозначений. Достоинством программируемых интегральных микросхем является наличие в одном корпусе таких устройств, которые могут выполнять много функций, и изменения или поправки могут быть внесены с помощью изменения выбранных кодов вместо перепрограммирования схем. К недостаткам программируемых интегральных микросхем относится отсутствие стандартов и сложность изучения и объяснения их работы.

В гл. 10 приведены типовые программы инициализации для устройств 8251, 8253 и 8255. Здесь же будет только кратко описано устройство 8255.

1.22.1. ПАРАЛЛЕЛЬНЫЙ ИНТЕРФЕЙС 8255 (ПРОГРАММИРУЕМЫЙ ПЕРИФЕРИЙНЫЙ ИНТЕРФЕЙС)

Программируемый периферийный интерфейс (PPI) 8255 содержит два 8-разрядных параллельных порта ввода-вывода (A и B) и два 4-разрядных порта (порт C, разряды 0 – 3 и разряды 4 – 7). Он имеет три основных режима работы, которые выбираются по содержанию регистра управления (в который можно только записывать), как показано на рис. 1.8. В табл. 1.9 описана адресация портов и управляющего регистра 8255.

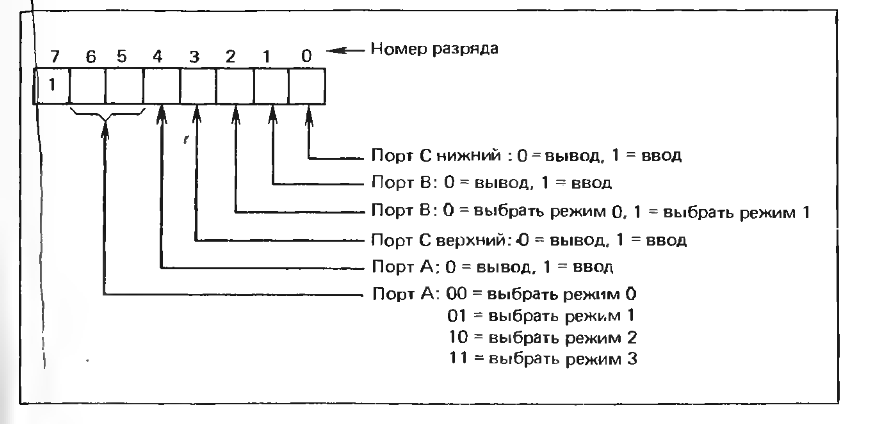


Рис. 1.8. Байты управления выбором режимов работы и направлений для параллельного интерфейса 8255

Таблица 1.9. *Адреса портов параллельного интерфейса 8255*

Адрес на входе		Выбираемый порт	Имя в примере
A_1	A_0		
0	0	Порт А	PORTA
0	1	Порт В	PORTB
1	0	Порт С	PORTC
1	1	Регистр управления*	CNTLP

* Заметим, что в регистр управления 8255 можно только записывать.

Основные особенности режимов работы состоят в следующем:

1. В режиме 0 все четыре порта работают независимо и пользователь может выбрать каждый из них в качестве порта ввода или вывода. При этом выводимые данные записываются в регистр-защелку, а вводимые – нет.

2. В режиме 1 три разряда порта С действуют как сигналы состояния и управления для порта А и три – аналогично для порта В. Это, по существу, режим работы "с рукопожатием". Если порт данных (А или В) является портом ввода, то сигналы состояния и управления определяются так, как показано на рис. 1.9 и описано далее:

STB (строб) – 0 с периферийного устройства загружает данные в регистр-защелку ввода;

IBF (входной буфер заполнен) – 1 указывает, что данные были загружены в регистр-защелку ввода, но процессор еще не прочитал их;

INTR (запрос прерывания) – 1 указывает, что данные были загружены в регистр-защелку ввода, но еще не прочитаны процессором, и что прерывание для порта разрешено. Этот сигнал может быть использован для прерывания работы процессора.

Если порт данных является портом вывода, то сигналы состояния и управления определяются следующим образом (см. рис. 1.10):

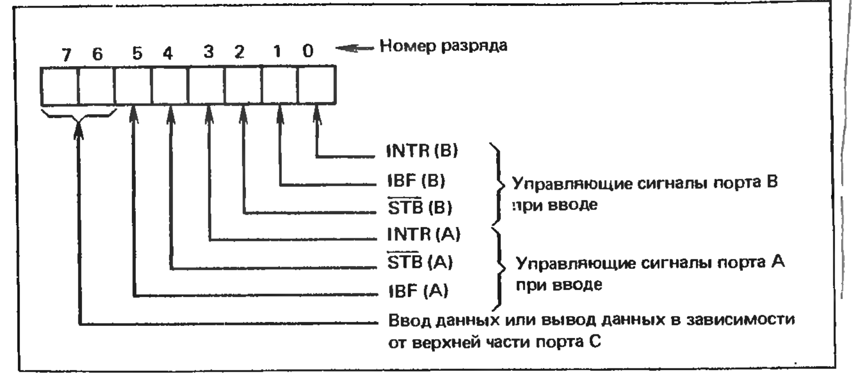


Рис. 1.9. Сигналы порта C 8255 при управлении вводом в режиме 1

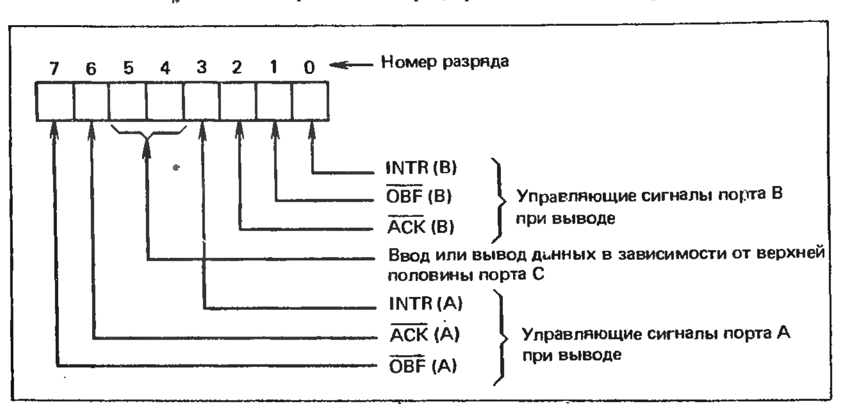


Рис. 1.10. Сигналы порта C 8255 при управлении выводом в режиме работы 1

ACK (подтверждение) – 0 (с периферийного устройства) указывает, что были получены самые последние выводившиеся данные и устройство готово для получения следующих. Таким образом, ACK служит в качестве сигнала готовности периферийного устройства;

OBF (буфер вывода заполнен) – 0 указывает, что процессор записал данные в порт, но периферийное устройство еще не получило их;

INTR (запрос прерывания) – 1 указывает, что периферийное устройство получило самые последние данные и готово для получения следующих. Кроме того, разрешено прерывание для порта. Этот сигнал может быть использован для прерывания работы процессора.

Каждая часть порта C также имеет разряд разрешения прерывания, который может быть установлен или сброшен. Установка разряда разрешает прерывания от порта данных (с помощью INTR), а очистка этого разряда запрещает их. Операции устройства 8255, управляемые прерываниями, будут рассматриваться позднее.

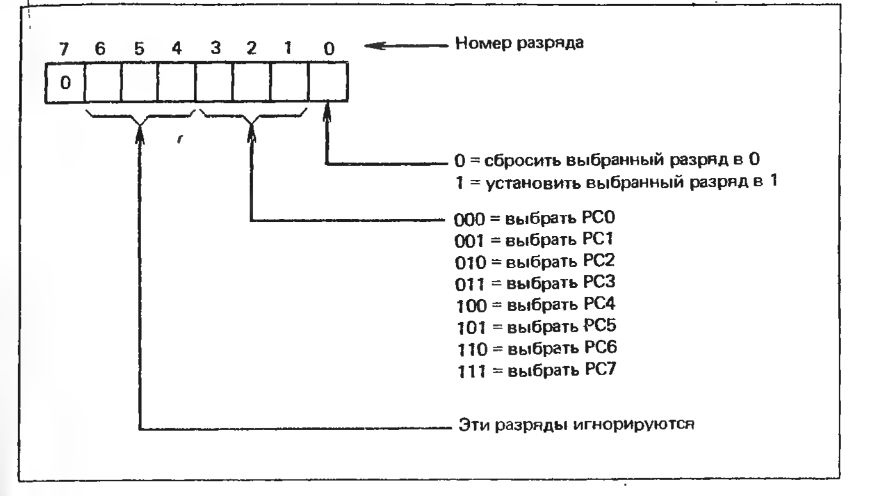


Рис. 1.11. Байты управления для установки или очистки разряда порта С 8255

3. В режиме 2 порт А является двунаправленным, а пять разрядов порта С действуют как управляющие сигналы для него. Этот режим не будет более рассматриваться, так как применяется не так часто, как режимы 0 и 1.

Пользователь должен понимать следующие особенности устройства 8255 при обычных режимах работы.

• При выборе направления передачи пользователь должен объявлять весь 4- или 8-разрядный порт портом ввода или портом вывода. Для этих целей нельзя выбирать отдельные разряды.

• Существует следующее соглашение для выбора направлений: 1 означает ввод, а 0 – вывод. При этом следует соблюдать осторожность, поскольку в других устройствах семейства 8080, 8085, таких как таймер ввода-вывода с произвольным доступом к памяти 8155, используется противоположное соглашение.

• Порт С позволяет устанавливать или очищать любой разряд с помощью посылки в порт управления специального командного байта (т. е. при выполнении команды OUT PORTC с разрядом данных 0, равным 0). Необходимый для этого формат показан на рис. 1.11. Эта процедура удобна для разрешения или запрещения прерываний и для изменения логических уровней линии управления, присоединенных к порту.

• Выводы записываются в регистр-защелку как в режиме 0, так и в режиме 1, а вводы – только в режиме 1.

• При сбросе все порты становятся портами ввода, регистр управления очищается и запрещаются прерывания.

• Содержимое регистра управления не может быть прочитано, так что его копия, если она необходима, должна храниться в ОЗУ.

В следующих примерах предполагается, что адреса для портов устройства 8255 даны в правой колонке табл. 1.9.

• Перевести все порты в режим работы 0. Назначить порт А портом ввода, порт В — портом вывода, разряды 0 — 3 порта С — для ввода, разряды 4 — 7 порта С — для вывода:

```
MVI  A,10010001B
OUT  CNTLP
```

Значения разрядов имеют следующий смысл:

разряд 7 = 1 — присвоить режимы и направления,
разряд 6 = 0 и разряд 5 = 0 — перевести порт А в режим 0,
разряд 4 = 1 — назначить порт А портом ввода,
разряд 3 = 0 — назначить разряды 4 — 7 порта С для вывода,
разряд 2 = 0 — перевести порт В в режим 0,
разряд 1 = 0 — назначить порт В портом вывода,
разряд 0 = 1 — назначить разряды 0 — 3 порта С для вывода.

Проверьте эти значения, обращаясь к рис. 1.8.

• Перевести порт А в режим 1 (режим работы с "рукопожатием"), а порт В — в режим 0. Назначить порт А портом вывода, порт В — портом ввода, разряды 0 — 3 порта С назначить для ввода, а разряды 4 — 7 порта С — для вывода:

```
MVI  A,10100011B
OUT  CNTLP
```

В данном случае, когда разряд 6 = 0 и разряд 5 = 1, порт А работает в режиме 1. Разряд 4 = 0 назначает порт А портом вывода, а разряд 1 = 1 назначает порт В портом ввода. Направления для порта С принимаются только по значению разрядов 0 — 2 (нижняя половина) и разрядов 4 — 5 (верхняя половина), так как разряды 3, 6 и 7 используются для сигналов "рукопожатие" порта А, как показано на рис. 1.10.

• Перевести порты А и В в режим работы 1 (работа "с рукопожатием"), назначить порт А портом ввода, порт В — портом вывода, а разряды 6 и 7 порта С назначить для вывода:

```
MVI  A,10110100B
OUT  CNTLP
```

Здесь разряд 4 = 1 назначает порт А портом ввода; разряд 2 = 1 переводит порт В в режим работы 1; разряд 1 = 1 назначает порт В портом ввода; направления передачи для порта С применяют только для разрядов 6 и 7, так как все разряды с 0 по 5, как показано на рис. 1.9 (порт А) и 1.10 (порт В), используются для сигналов "рукопожатия".

• Ожидать сигнал IBF для порта А и затем прочитать данные из порта А и записать их в ячейку памяти INDAT. Предполагается, что порт А является портом ввода в режиме 1:

```
WAITR:  IN    PORTC        ;ВЫХОДНЫЕ ДАННЫЕ ДОСТУПНЫ?
        ANI   00100000B
        JZ    WAITR        ;НЕТ, ЖДАТЬ
        IN    PORTA        ;ДА, ПРОЧИТАТЬ ДАННЫЕ
        STA   INDAT        ;СОХРАНИТЬ ДАННЫЕ В ПАМЯТИ
```

Если порт А является портом ввода в режиме работы 1, то сигнал IBF порта А находится в разряде 5 порта С (см. рис. 1.9).

• Ожидать сигнала "подтверждение" для порта B, а затем послать данные из ячейки памяти OUTDAT в порт B. Предполагается, что порт B является портом вывода в режиме работы 1:

```
WAITR:    IN    PORTC         ;ПЕРИФЕРИЙНОЕ УСТРОЙСТВО ГОТОВО?
          ANI   00000100B
          JNZ   WAITR         ;НЕТ, ЖДАТЬ
          LDA   OUTDAT        ;ДА, ВЗЯТЬ ДАННЫЕ ИЗ ПАМЯТИ
          OUT   PORTB         ;ПОСЛАТЬ ДАННЫЕ В ПОРТ ВЫВОДА
```

Если порт B является портом вывода в режиме работы 1, то сигнал "подтверждение" для порта B (в периферийное устройство готово) находится в разряде 2 порта C (см. рис. 1.10).

• Установить разряд 5 порта C в 1 без изменения остальных разрядов:

```
          MVI   A,00001011B   ;УСТАНОВИТЬ РАЗРЯД 5 ПОРТА C
          OUT   CNTLP
```

Здесь разряд 7, равный 0, служит для выполнения операции установки или сброса разряда, разряды 3, 2 и 1, равные 101_2 (5_{10}), служат для выбора разряда 5 порта C, а разряд 0, равный 1, – для установки разряда. Вы можете проверить эти значения разрядов по рис. 1.11.

• Очистить разряд 0 порта C без изменения остальных разрядов:

```
          SUB   A             ;ОЧИСТИТЬ РАЗРЯД 0 ПОРТА C
          OUT   CNTLP
```

Здесь разряды 3, 2 и 1 равны 000_2 (0_{10}) для выбора разряда 0 порта C, а разряд 0 = 0 – для очистки разряда.

Отметим следующие проблемы, встречающиеся при использовании устройства 8255:

• Когда порт A или порт B назначается как порт вывода, на отдельных вводах интегральной микросхемы может быть сигнал высокого или низкого уровня, заданный при записи данных в порт ввода-вывода до назначения направления передачи. Когда одна или обе половины порта C назначены как порт вывода, уровни всех сигналов будут заведомо низкими.

• При работе в режиме 0 избегайте назначения верхней и нижней половин порта C для ввода и вывода. Некоторые пользователи сообщали, что чтение из порта C может изменять уровни выходных сигналов, а запись в порт C может изменять буфер ввода и, таким образом, модифицировать значения разрядов, установленные входным сигналом, причем это происходит не всегда.

• При режимах работы 1 и 2 управляющий байт, показанный на рис. 1.11, должен использоваться для записи в порт C по одному разряду за раз. Байт не может быть просто записан прямо в порт C. Кроме того, при режимах работы 1 и 2 сигналы управления должны быть сначала инициализированы с помощью записи соответствующих разрядов в порт C с использованием соответствующих управляющих байтов.

1.23. НАПИСАНИЕ ПРОГРАММ, РАБОТАЮЩИХ ПО ПРЕРЫВАНИЯМ

В большинстве систем прерывания 8080 и 8085 [6, 7] используются команды RST и входные сигналы, которые передают управление по определенным адресам памяти, перечисленным в табл. 1.10. Все команды RST и входные сигналы сохраняют старое значение программного счетчика в вершине стека,

Таблица 1.10. *Команды регистра и входные сигналы*

Мнемоническая форма	Двоичная форма	Шестнадцатеричная форма	Адрес назначения (шестнадцатеричный)
RST 0	11000111	C7	0000
RST 1	11001111	CF	0008
RST 2	11010111	D7	0010
RST 3	11011111	DF	0018
RST 4	11100111	E7	0020
RST 5	11101111	EF	0028
RST 6	11110111	F7	0030
RST 7	11111111	FF	0038
TRAP (RST 4.5)			0024
RST 5.5*			002C
RST 6.5*			0034
RST 7.5*			003C

* Только в 8085 (раздельные вводы)

но не сохраняют автоматически остальные регистры. Все регистры пользователя сохраняются при такой последовательности команд:

```
PUSH PSW            ;СОХРАНИТЬ ВСЕ РЕГИСТРЫ
PUSH B
PUSH D
PUSH H
```

Слово состояния процессора (PSW) содержит аккумулятор (старший байт) и флаги (младший байт). Обратная последовательность восстанавливает регистры пользователя:

```
POP  H              ;ВОССТАНОВИТЬ РЕГИСТРЫ
POP  D
POP  B
POP  PSW
```

Прерывания должны быть разрешены явно с помощью команды EI непосредственно перед командой RET, заканчивающей обслуживающую программу. Команда EI задерживает действительное разрешение прерываний на один такт команды во избежание лишней записи адреса возврата в стек (т. е. команда RET может извлечь из стека адрес возврата до того, как будет распознано ожидающее обработку прерывание).

Если в конце программы необходимо восстанавливать какие-либо регистры, допускающие только запись в них, то при сохранении этих регистров будьте особенно внимательны. Например, блок управления приоритетными прерываниями 8214 имеет такой регистр. Копию содержимого этого регистра необходимо сохранять в оперативной памяти и восстанавливать из стека:

```
PUSH PSW       ;СОХРАНИТЬ ВСЕ РЕГИСТРЫ
PUSH B
PUSH D
PUSH H
LDA  PRTY      ;СОХРАНИТЬ СТАРЫЙ ПРИОРИТЕТ
PUSH PSW
MVI  A,NPRTY   ;ВЗЯТЬ НОВЫЙ ПРИОРИТЕТ
OUT  PPORT     ;ПОСЛАТЬ ЕГО ВО ВНЕШНИЙ РЕГИСТР ПРИОРИТЕТА
STA  PRTY      ;ЗАПИСАТЬ КОПИЮ НОВОГО ПРИОРИТЕТА В ОЗУ
```

Процедура должна возвращать как предыдущий приоритет, так и исходное содержимое регистров:

```
        POP  PSW        ;ВОССТАНОВИТЬ СТАРЫЙ ПРИОРИТЕТ
        OUT  PPORT      ;ПОСЛАТЬ ЕГО ВО ВНЕШНИЙ РЕГИСТР ПРИОРИТЕТА
        STA  PRTY       ;СОХРАНИТЬ КОПИЮ ПРИОРИТЕТА В ОЗУ
        POP  H          ;ВОССТАНОВИТЬ РЕГИСТРЫ
        POP  D
        POP  B
        POP  PSW
```

Для повторной используемости программы все временные данные следует записывать в стек после записи в него содержимого регистров. Как отмечено при обсуждении методов передачи параметров, область памяти в стеке (NTEMP байт) можно назначить с помощью последовательности команд:

```
        LXI  H,-NTEMP   ;НАЗНАЧИТЬ NTEMP СВОБОДНЫХ БАЙТОВ В СТЕКЕ
        DAD  SP
        SPHL            ;ОБЫЧНЫЙ СПОСОБ КОРРЕКТИРОВКИ SP
```

Разумеется, потом эта область хранения данных должна быть удалена с помощью последовательности команд

```
        LXI  H,NTEMP    ;УДАЛИТЬ NTEMP ПУСТЫХ БАЙТОВ ИЗ СТЕКА
        DAD  SP
        SPHL            ;ОБЫЧНЫЙ СПОСОБ КОРРЕКТИРОВКИ SP
```

Если число NTEMP мало, то следует заменить эту последовательность на NTEMP команд DCX SP или INX SP для сокращения времени работы и памяти. В гл. 11 приведены примеры простых программ обслуживания прерываний.

1.23.1. ПРЕРЫВАНИЯ УСТРОЙСТВА 8255

Чтобы использовать параллельный интерфейс 8255 в системе, управляемой прерываниями, на нем должен быть установлен разряд разрешения прерываний. На рис. 1.11 описаны выходные величины, которые должна установить программа на доступном только для записи регистре управления устройства 8255. Прерывания для порта В разрешает разряд 2, а для порта А – в зависимости от его назначения (для ввода или вывода) разряд 4 или 6. Установка разряда разрешения прерываний разрешает прерывания; очистка этого разряда запрещает прерывания. В следующих примерах предполагается, что устройство 8255 работает в режиме 1, а его регистром управления является порт вывода CNTLP.

Примеры

1. Разрешить прерывания при вводе из порта А:

```
        MVI  A,00001001B
        OUT  CNTLP
```

Разряд 7 = 0 для выполнения функции установка-сброс; разряды 3, 2 и 1 равны 100_2 (4_{10}) для выбора разряда 4 (разрешить прерывания, когда порт А является портом ввода); разряд 0 = 1 для установки разряда разрешения и, таким образом, для разрешения прерываний. Разряды 4,5 и 6 не используются.

2. Разрешить прерывания при выводе из порта В:

```
        MVI  A,00000101B
        OUT  CNTLP
```

Разряды 3, 2 и 1 равны 010_2 (2_{10}) для выбора разряда 2 (разрешить прерывания при выводе из порта В).

3. Запретить прерывания при выводе из порта А:

```
MVI   A,00001100B
OUT   CNTLP
```

Разряды 3, 2 и 1 равны 110_2 (6_{10}) для выбора разряда 6 (разрешить прерывания, когда порт А является портом вывода), а разряд 0 = 0 для очистки разряда разрешения, и, таким образом, запрещения прерываний.

Для порта вывода с разрешенным прерыванием прерывание возникает после того, как периферийное устройство явно получит самые последние данные (если до этого вывода не было). Таким образом, обычно на устройство вывода необходимо послать нуль, чтобы перевести его в неактивное состояние (иначе говоря, для начала запретить прерывания от него). Для портов ввода этой проблемы нет, так как изначально буфер ввода заведомо пустой.

Когда схема прерываний устройства 8255 находится в состоянии "прерывания разрешены" и задан режим работы 1 или 2, процессор может посылать запросы прерывания с помощью сигналов INTRA и INTRB (соответственно выходы 3 и 0 порта ввода-вывода С). Запросы прерывания появятся только в том случае, если разряды с помощью байта управления были установлены в 1 (см. рис. 1.11).

1.24. УВЕЛИЧЕНИЕ БЫСТРОДЕЙСТВИЯ ПРОГРАММ

Ускорить выполнение можно, вообще говоря, только определив, где теряется время [8]. Для этого необходимо определить, какие циклы процессор выполняет наиболее часто (это не относится к специальным программам задержки). Основное влияние на снижение затрат времени часто выполняемого цикла оказывает счетчик числа циклов. Таким образом, важно определить, как часто выполняются команды, и работать далее с циклами в порядке частоты их выполнения.

После того, как уточнено, какие циклы выполняются наиболее часто, время их выполнения можно уменьшать, используя следующие приемы.

• Исключайте избыточные операции. К ним относится добавление констант на каждой итерации или проверка каких-либо специальных случаев на каждой итерации. Сюда могут входить также выбор из памяти константы и адреса каждый раз вместо их однократной записи в регистр или пару регистров.

• Реорганизуйте цикл так, чтобы уменьшить число команд перехода. Переходы часто можно исключить, изменив начальные условия, порядок операций или комбинируя несколько операций. В частности, может быть полезно все начинать на шаг раньше, делая, таким образом, первую операцию такой же, как и все остальные. Если осуществляется сравнение чисел, то может оказаться полезным изменить порядок операций на обратный, так как при этом случай равенства можно не обрабатывать отдельно. Переорганизация может также позволить пользователю комбинировать проверку условий внутри цикла с общим управлением циклом.

• Используйте линейную последовательность команд вместо подпрограмм. Это исключит, по крайней мере, команды CALL и RETURN.

• Используйте для временного хранения стек вместо определенных адресов памяти. Напомним, что команда XTHL обменивает вершину стека с регистрами H и L и, таким образом, может восстанавливать старое значение и одновременно сохранять текущее значение.

• Назначайте регистры таким образом, чтобы получить максимальный выигрыш от таких специализированных команд, как LHLD, SHLD, XCHG, XTHL и PCHL. В частности, всегда старайтесь помещать адрес в пару регистров D (так как есть команда XCHG), а не в пару регистров B.

• Везде, где возможно, для работы с 16-разрядными данными используйте 16-разрядные команды (LHLD, SHLD, INX, DCX, DAD, PUSH, POP, XCHG, XTHL, PCHL и SPHL).

• Используйте команды MVI M, INR M и DCR M для работы с данными в памяти; при этом нет необходимости сохранять и восстанавливать регистры.

• Используйте команды MOV, MVI, INR, DCR, INX, DCX, SHLD, LHLD, XCHG, XTHL, PUSH, POP, PCHL и SPHL для работы с данными в регистрах; при этом нет необходимости сохранять и восстанавливать аккумулятор.

• Используйте RST, PCHL или RET в качестве коротких команд перехода. Требуемые при этом адреса, разумеется, либо не должны использоваться (RST), либо все же могут потребоваться (PCHL или RET) для других целей. Заметим, что если Вам нужно загрузить адрес назначения в H и L или же записать его в стек, то на выполнение команд PCHL и RET затрачивается больше времени, однако затраты времени уменьшаются, если адрес назначения уже находится в нужном месте.

• Организуйте последовательности условных переходов таким образом, чтобы минимизировать среднее время выполнения. Осуществляемые часто переходы должны стоять перед теми, которые случаются редко. Например, проверка результата на знак "минус" (если величина случайная, то это условие проверки удовлетворяется для 50 % случаев) должна стоять перед проверкой на 0 (для случайной величины верно в 1 % случаев).

• Проверяйте на условия, при которых последовательность команд не выполняется, и обходите эту последовательность для случая выполнения условий. Это будет выгодно, если последовательность команд длинная и не изменяет результата. Типичным примером является распространение переносов через байты более высокого порядка. Если перенос появляется редко, то в среднем будет быстрее проверить на перенос, чем просто распространять 0.

Обычный путь к снижению времени выполнения состоит в замене длинных последовательностей команд таблицами. Если не предусмотрены специальные выходы или не введены элементы логики программы, то с помощью отдельного просмотра таблицы могут выполняться те же операции, что и с помощью последовательности команд. Ценой этого является необходимость в расходах на дополнительную память, но при ее наличии это может быть оправдано. Если емкость памяти достаточна, то поиск в таблице может быть рациональным подходом, даже если многие записи ее одинаковы. Кроме того, что ускоряется работа, поиск в таблице обычно легче программировать и проще изменять.

Длину программы можно значительно уменьшить, выделив общие последовательности команд и заменив их подпрограммами. В результате получается одна копия этих команд вместо нескольких копий; правда, при этом увеличивается время выполнения из-за наличия команд CALL и RET. Чем больше команд можно заменить на подпрограммы, тем больше экономится памяти. Конечно, такие подпрограммы обычно не являются общими и могут быть трудными для понимания и использования. Некоторые последовательности могут уже быть в мониторе или другой системной программе; в этом случае такие последовательности могут быть заменены на обращения к системной программе, если только возврат правильно выполняется.

Некоторые методы уменьшения времени выполнения снижают также и размер используемой памяти. В частности, удаление избыточных операций, реорганизация циклов, использование стека, упорядочение использования регистров, использование 16-разрядных команд, работа непосредственно с памятью или регистрами и применение специальных команд переходов уменьшают как размер используемой памяти, так и время выполнения. Конечно, применение линейной последовательности команд вместо циклов и подпрограмм уменьшает время выполнения, но увеличивает длину программы.

Поиск по таблице обычно требует большего объема памяти, но снижает время выполнения. Снизить эти требования к памяти можно, исключая промежуточные значения и интерполируя результаты, исключая избыточные данные с помощью специальных проверок и уменьшая порядок входных величин [9, 10]. Часто некоторые предварительные проверки и ограничения могут значительно уменьшить размер необходимой таблицы.

ГЛАВА 2

РЕАЛИЗАЦИЯ ДОПОЛНИТЕЛЬНЫХ КОМАНД И СПОСОБОВ АДРЕСАЦИИ

В этой главе показано, как реализовать команды и способы адресации, которые не входят в набор команд 8080 или 8085. Конечно, нет набора команд, который включал бы все возможные комбинации. Разработчик должен выбирать набор команд, исходя из того, сколько кодов операций доступно, насколько легко могут быть выполнены дополнительные комбинации и как часто они могут использоваться. Описание дополнительных команд и способов адресации не означает, что основной набор команд является неполным или же плохо разработанным.

Наше внимание будет обращено на дополнительные команды и способы адресации, которые:

аналогичны командам и способам адресации, включенным в набор команд, описаны в работах [1, 11],

обсуждаются в работе [3],

выполняются в других микропроцессорах, особенно в тех, которые похожи на рассматриваемые или частично сравнимы с ними [12].

Эта глава, по-видимому, будет представлять особый интерес для тех, кто близко знаком с языками ассемблера других ЭВМ.

2.1. РАСШИРЕНИЯ НАБОРА КОМАНД

Расширения набора команд описываются здесь в соответствии с системой, предлагаемой в проекте стандарта IEEE Task P 694 [1]. Команды делятся на следующие группы (перечисленные в порядке рассмотрения в данной главе): арифметические, логические, передачи данных, перехода, пропуска, вызова подпрограммы, возврата из подпрограммы и смешанные. Типы операндов для каждого типа команд обсуждаются в следующем порядке: байт (8-разрядный), слово (16-разрядное), десятичный операнд, разряд, число, составной операнд. При обсуждении способов адресации используется следующий порядок: прямая, косвенная, непосредственная, индексная, регистровая, автоиндексирование с предварительным увеличением адреса, автоиндексирование с последующим увеличением адреса, автоиндексирование с предварительным уменьшением адреса, автоиндексирование с последующим уменьшением адреса[1], косвенная с предварительным индексированием (называемая также предындексированной или индексированной косвенно), и косвенная с последующим индексированием (называемая также послеиндексированной или косвенной индексированной).

2.2. АРИФМЕТИЧЕСКИЕ КОМАНДЫ

В эту группу включены следующие команды: сложение, сложение с флагом переноса, вычитание, вычитание при перестановке операндов, вычитание с флагом переноса (заем), увеличение на 1, уменьшение на 1, умножение, деление, сравнение, получение дополнения до двух (отрицательного числа) и расширение. Для удобства те команды, принадлежность которых к конкретной категории неясна, повторяются во всех категориях, к которым они могли бы быть отнесены.

2.2.1. КОМАНДЫ СЛОЖЕНИЯ (БЕЗ ФЛАГА ПЕРЕНОСА)

1. Добавить ячейку памяти ADDR к аккумулятору:

```
LXI  H,ADDR     ;ВЗЯТЬ АДРЕС ДАННЫХ
ADD  M          ;ЗАТЕМ ПРИБАВИТЬ ДАННЫЕ
```

2. Добавить флаг переноса к аккумулятору:

```
ACI  0          ;(A) = (A) + ФЛАГ ПЕРЕНОСА
```

3. Добавить в десятичном виде к аккумулятору флаг переноса:

```
ACI  0          ;(A) = (A) + ФЛАГ ПЕРЕНОСА
DAA             ;КОРРЕКТИРОВАТЬ В ДЕСЯТИЧНЫЙ ВИД
```

4. Добавить в десятичном виде к аккумулятору число VALUE:

```
ADI  VALUE      ;(A) = (A) + VALUE
DAA             ;КОРРЕКТИРОВАТЬ В ДЕСЯТИЧНЫЙ ВИД
```

5. Добавить в десятичном виде к аккумулятору регистр:

```
ADD  REG        ;(A) = (A) + (REG)
DAA             ;КОРРЕКТИРОВАТЬ В ДЕСЯТИЧНЫЙ ВИД
```

---

[1] Далее в книге эти способы автоиндексирования будем называть соответственно предувеличением, послеувеличением, предуменьшением, послеуменьшением. (*Прим. перев.*)

6. Добавить 16-разрядное число к паре регистров HL:

```
LXI  RP,VAL16
DAD  RP          ;HL = HL + VAL16
```

Здесь в качестве RP может быть пара регистров B и C или D и E.

7. Добавить к паре регистров HL ячейки памяти ADDR и ADDR + 1 (старший байт в ADDR + 1) :

```
XCHG             ;1-Й ОПЕРАНД В DE
LHLD ADDR        ;ВЫБРАТЬ 2-Й ОПЕРАНД
DAD  D           ;СЛОЖИТЬ ОПЕРАНДЫ
```

В памяти операнд запоминается в обычном для 8080, 8085 формате, при котором младший байт находится по меньшему адресу.

8. Добавить ячейки памяти ADDR и ADDR + 1 (старший байт в ADDR + 1) к ячейкам памяти SUM и SUM + 1 (старший байт в SUM + 1) :

```
LHLD SUM         ;ВЗЯТЬ ТЕКУЩЕЕ ЗНАЧЕНИЕ СУММЫ
XCHG
LHLD ADDR        ;ДОБАВИТЬ ЭЛЕМЕНТ
DAD  D
SHLD SUM         ;СОХРАНИТЬ ОБНОВЛЕННУЮ СУММУ
```

9. Добавить 16-разрядное число VAL16 к ячейкам памяти ADDR и ADDR + 1 (старший байт в ADDR + 1) :

```
LHLD ADDR        ;ВЗЯТЬ ТЕКУЩЕЕ ЗНАЧЕНИЕ СУММЫ
LXI  RP,VAL16    ;ДОБАВИТЬ ЭЛЕМЕНТ
DAD  RP
SHLD ADDR        ;СОХРАНИТЬ ОБНОВЛЕННУЮ СУММУ
```

Здесь в качестве RP может быть пара регистров B и C или D и E.

2.2.2. КОМАНДЫ СЛОЖЕНИЯ (С ФЛАГОМ ПЕРЕНОСА)

1. Добавить ячейку памяти ADDR и флаг переноса к аккумулятору:

```
LXI  H,ADDR      ;ВЗЯТЬ АДРЕС ДАННЫХ
ADC  M           ;ЗАТЕМ ПРИБАВИТЬ ДАННЫЕ И ФЛАГ ПЕРЕНОСА
```

2. Добавить флаг переноса к аккумулятору:

```
ACI  0           ;(A) = (A) + ФЛАГ ПЕРЕНОСА
```

3. Добавить в десятичном виде к аккумулятору VALUE и флаг переноса:

```
ACI  VALUE       ;(A) = (A) + VALUE + ФЛАГ ПЕРЕНОСА
DAA              ;КОРРЕКТИРОВАТЬ В ДЕСЯТИЧНЫЙ ВИД
```

4. Добавить в десятичном виде к аккумулятору регистр и флаг переноса:

```
ADC  REG         ;(A) = (A) + REG + ФЛАГ ПЕРЕНОСА
DAA              ;КОРРЕКТИРОВАТЬ В ДЕСЯТИЧНЫЙ ВИД
```

5. Добавить пару регистров и флаг переноса к паре регистров HL:

```
MOV  A,L         ;СЛОЖИТЬ МЛАДШИЕ БАЙТЫ
ADC  RPL
MOV  L,A
```

```
        MOV   A,H         ;ЗАТЕМ СЛОЖИТЬ СТАРШИЕ БАЙТЫ
        ADC   RPH
        MOV   H,A
```

или

```
        JNC   NOCRY       ;ФЛАГ ПЕРЕНОСА РАВЕН 1?
        INX   H           ;ДА, ДОБАВИТЬ ЕЩЕ 1
NOCRY:  DAD   RP          ;ДОБАВИТЬ 16 РАЗРЯДОВ
```

Здесь RPH – старший байт пары регистров RP, а RPL – младший байт.

2.2.3. КОМАНДЫ ВЫЧИТАНИЯ (БЕЗ ЗАЕМА)

1. Вычесть ячейку памяти ADDR из аккумулятора:

```
        LXI   H,ADDR      ;ВЗЯТЬ АДРЕС ДАННЫХ
        SUB   M           ;ЗАТЕМ ВЫЧЕСТЬ ДАННЫЕ
```

2. Вычесть занятый ранее разряд (флаг переноса) из аккумулятора:

```
        SBI   0           ;A = A - ФЛАГ ПЕРЕНОСА
```

3. Вычесть в десятичном виде из аккумулятора VALUE:

```
        MOV   REG,A       ;СОХРАНИТЬ УМЕНЬШАЕМОЕ
        MVI   A,9AH       ;ВЫЧИСЛИТЬ 100 - ВЫЧИТАЕМОЕ
        SUI   VALUE
        ADD   REG         ;ПРИБАВИТЬ УМЕНЬШАЕМОЕ
        DAA               ;КОРРЕКТИРОВАТЬ РЕЗУЛЬТАТ В ДЕСЯТИЧНЫЙ ВИД
```

Флаг переноса является инвертированным заемом, т. е. флаг переноса = 1, если вычитание не требует заема, и 0 – в противном случае.

4. Вычесть в десятичном виде регистр из аккумулятора:

```
        MOV   REG1,A      ;СОХРАНИТЬ УМЕНЬШАЕМОЕ
        MVI   A,9AH       ;ВЫЧИСЛИТЬ 100 - ВЫЧИТАЕМОЕ
        SUB   REG
        ADD   REG1        ;ПРИБАВИТЬ УМЕНЬШАЕМОЕ
        DAA               ;КОРРЕКТИРОВАТЬ РЕЗУЛЬТАТ В ДЕСЯТИЧНЫЙ ВИД
```

Флаг переноса является инвертированным заемом.

5. Вычесть 16-разрядное число из пары регистров HL:

```
        LXI   RP,-VAL16 ;HL = HL - VAL16
        DAD   RP
```

Флаг переноса является инвертированным заемом.

6. Вычесть пару регистров из пары регистров HL.

```
        MOV   A,L         ;ВЫЧЕСТЬ МЛАДШИЕ БАЙТЫ
        SUB   RPL
        MOV   L,A
        MOV   A,H         ;ВЫЧЕСТЬ СТАРШИЕ БАЙТЫ С ЗАЕМОМ
        SBB   RPH
        MOV   H,A
```

2.2.4. КОМАНДЫ ВЫЧИТАНИЯ ПРИ ПЕРЕСТАНОВКЕ ОПЕРАНДОВ

1. Вычесть аккумулятор из VALUE и поместить разность в аккумулятор:

```
        CMA               ;ПОЛУЧИТЬ ОТРИЦАТЕЛЬНОЕ ЗНАЧЕНИЕ A
```

```
        INR  A
        ADI  VALUE     ;СФОРМИРОВАТЬ  - A + VALUE
```

или

```
        MOV  REG,A     ;ВЫЧИСЛИТЬ VALUE - A
        MVI  A,VALUE
        SUB  REG
```

Флаг переноса является инвертированным заемом в первом методе и заемом — во втором.

2. Вычесть аккумулятор из регистра и поместить разность в аккумулятор:

```
        CMA            ;ПОЛУЧИТЬ ОТРИЦАТЕЛЬНОЕ ЗНАЧЕНИЕ A
        INR  A
        ADD  REG       ;СФОРМИРОВАТЬ  - A + REG
```

или

```
        MOV  REG1,A    ;ВЫЧИСЛИТЬ REG - A
        MOV  A,REG
        SUB  REG1
```

В первом методе флаг переноса является инвертированным заемом, а во втором — заемом.

3. Вычесть в десятичном виде аккумулятор из VALUE и поместить разность в аккумулятор:

```
        MOV  REG,A
        MVI  A,9AH     ;ВЫЧИСЛИТЬ 100 - ВЫЧИТАЕМОЕ
        SUB  REG
        ADI  VALUE     ;ПРИБАВИТЬ УМЕНЬШАЕМОЕ
        DAA            ;КОРРЕКТИРОВАТЬ РЕЗУЛЬТАТ В ДЕСЯТИЧНЫЙ ВИД
```

4. Вычесть в десятичном виде аккумулятор из регистра и поместить разность в аккумулятор:

```
        MOV  REG1,A
        MVI  A,9AH     ;ВЫЧИСЛИТЬ 100 - ВЫЧИТАЕМОЕ
        SUB  REG1
        ADD  REG       ;ПРИБАВИТЬ УМЕНЬШАЕМОЕ
        DAA            ;КОРРЕКТИРОВАТЬ РЕЗУЛЬТАТ В ДЕСЯТИЧНЫЙ ВИД
```

2.2.5. КОМАНДЫ ВЫЧИТАНИЯ С ЗАЕМОМ (ФЛАГОМ ПЕРЕНОСА)

1. Вычесть ячейку памяти ADDR из аккумулятора с заемом:

```
        LXI  H,ADDR    ;ВЗЯТЬ АДРЕС ДАННЫХ
        SBB  M         ;ЗАТЕМ ВЫЧЕСТЬ ДАННЫЕ С ЗАЕМОМ
```

2. Вычесть заем (флаг переноса) из аккумулятора:

```
        SBI  0
```

3. Вычесть в десятичном виде инвертированный заем (флаг переноса = 1, если перед этим не было заема, и 0 — в противном случае) из аккумулятора:

```
        ACI  99H       ;ДОБАВИТЬ 99 И ФЛАГ ПЕРЕНОСА
        DAA            ;СДЕЛАТЬ РЕЗУЛЬТАТ ДЕСЯТИЧНЫМ
```

Если при вычитании был заем, то флаг переноса в конце работы равен 0, а в противном случае — 1.

4. Вычесть в десятичном виде из аккумулятора VALUE и инвертированный заем:

```
MOV  REG,A       ;СОХРАНИТЬ УМЕНЬШАЕМОЕ
MVI  A,99H       ;ВЫЧИСЛИТЬ
ACI  0           ; 100 - ВЫЧИТАЕМОЕ - (1-ФЛАГ ПЕРЕНОСА)
SUI  VALUE
ADD  REG         ;ПРИБАВИТЬ УМЕНЬШАЕМОЕ
DAA              ;СДЕЛАТЬ РЕЗУЛЬТАТ ДЕСЯТИЧНЫМ
```

Флаг переноса является инвертированным заемом.

5. Вычесть в десятичном виде из аккумулятора регистр и инвертированный заем:

```
MOV  REG1,A      ;СОХРАНИТЬ УМЕНЬШАЕМОЕ
MVI  A,99H       ;ВЫЧИСЛИТЬ
ACI  0           ; 100 - ВЫЧИТАЕМОЕ - (1-ФЛАГ ПЕРЕНОСА)
SUB  REG
ADD  REG1        ;ПРИБАВИТЬ УМЕНЬШАЕМОЕ
DAA              ;СДЕЛАТЬ РЕЗУЛЬТАТ ДЕСЯТИЧНЫМ
```

6. Вычесть 16-разрядное число и заем из пары регистров HL:

```
        JNC  NOCRY      ;СФОРМИРОВАТЬ HL - ФЛАГ ПЕРЕНОСА
        DCX  H
NOCRY:  LXI  RP,-VALUE ;HL = HL - ФЛАГ ПЕРЕНОСА - VALUE
        DAD  RP
```

В конце работы флаг переноса является инвертированным заемом.

7. Вычесть пару регистров и заем из пары регистров HL:

```
MOV  A,L         ;ВЫЧЕСТЬ МЛАДШИЕ БАЙТЫ С ЗАЕМОМ
SBB  RPL
MOV  L,A
MOV  A,H         ;ВЫЧЕСТЬ СТАРШИЕ БАЙТЫ С ЗАЕМОМ
SBB  RPH
MOV  H,A
```

2.2.6. КОМАНДЫ УВЕЛИЧЕНИЯ НА 1

1. Увеличить на 1 ячейку памяти ADDR:

```
LXI  H,ADDR
INR  M
```

2. Увеличить на 1 аккумулятор, установив флаг переноса, если результат равен 0:

```
ADI  1
```

Напомним, что команда INR не изменяет флага переноса, хотя и влияет на флаг нуля.

3. Увеличить в десятичном виде аккумулятор на 1 (добавить 1 к A в десятичном виде):

```
ADI  1
DAA
```

Здесь нельзя использовать команду INR, так как она не изменяет флага переноса.

4. Увеличить на 1 ячейки памяти ADDR и ADDR + 1 (старший байт в ADDR + 1):

```
            LHLD ADDR
            INX  H          ;16-РАЗРЯДНОЕ УВЕЛИЧЕНИЕ НА 1
            SHLD ADDR
```

или

```
            LXI  H,ADDR
            INR  M          ;ДОБАВИТЬ 1 К МЛАДШЕМУ БАЙТУ
            JNZ  DONE
            INX  H          ;ДОБАВИТЬ 1 К СТАРШЕМУ БАЙТУ
            INR  M
            DCX  H
 DONE:      NOP
```

Второй метод оставляет ADDR для последующего использования в регистрах H и L.

5. Увеличить на 1 пару регистров, установив флаг нуля, если результат равен 0:

```
            INX  RP         ;16-РАЗРЯДНОЕ УВЕЛИЧЕНИЕ НА 1
            MOV  A,RPL      ;ПРОВЕРИТЬ 16-РАЗРЯДНЫЙ РЕЗУЛЬТАТ НА НУЛЬ
            ORA  RPH
```

Эта последовательность команд изменяет аккумулятор и остальные флаги (напомним, что ORA очищает флаг переноса).

2.2.7. КОМАНДЫ УМЕНЬШЕНИЯ НА 1

1. Уменьшить на 1 ячейку памяти ADDR:

```
            LXI  H,ADDR
            DCR  M
```

2. Уменьшить на 1 аккумулятор и, если был заем, установить флаг переноса:

```
            SUI  1
```

3. Уменьшить на 1 аккумулятор и, если не было заема, установить флаг переноса:

```
            ADI  0FFH
```

4. Уменьшить в десятичном виде аккумулятор на 1 (вычесть в десятичном виде 1 из аккумулятора):

```
            ADI  99H        ;ДЛЯ ВЫЧИТАНИЯ 1 ДОБАВИТЬ ДЕСЯТИЧНОЕ ЧИСЛО 99
            DAA
```

Флаг переноса здесь является инвертированным заемом; он очищается, если был перенос, и устанавливается в противном случае.

5. Уменьшить на 1 ячейки памяти ADDR и ADDR + 1 (старший байт в ADDR + 1)

```
            LHLD ADDR
            DCX  H          ;16-РАЗРЯДНОЕ УМЕНЬШЕНИЕ НА 1
            SHLD ADDR
```

6. Уменьшить на 1 пару регистров, установив флаг нуля, если результат равен 0

```
DCX  RP        ;16-РАЗРЯДНОЕ УМЕНЬШЕНИЕ НА 1
MOV  A,RPL     ;ПРОВЕРИТЬ ПАРУ РЕГИСТРОВ НА НУЛЬ
ORA  RPH
```

Эта последовательность команд затирает старое значение аккумулятора и изменяет остальные флаги (напомним, что команда ORA очищает флаг переноса).

2.2.8. КОМАНДЫ УМНОЖЕНИЯ

1. Умножить аккумулятор на 2:

```
ADD  A
```

2. Умножить аккумулятор на 3 (используя для временного хранения REG)

```
MOV  REG,A     ;СОХРАНИТЬ A
ADD  A         ;2 X A
ADD  REG       ;3 X A
```

3. Умножить аккумулятор на 4:

```
ADD  A         ;2 X A
ADD  A         ;4 X A
```

Примеры 1, 2 и 3 легко можно расширить для умножения на другие небольшие целые числа.

4. Умножить регистры H и L на 2:

```
DAD  H
```

Этот прием умножения на небольшие целые числа дает 16-разрядный результат.

5. Умножить регистры H и L на 3 (используя для временного хранения RP):

```
MOV  RPH,H
MOV  RPL,L
DAD  H         ;2 X HL
DAD  RP        ;3 X HL
```

Заметим, что команда XCHG не может быть здесь использована, так как она затирает старое содержимое пары регистров HL.

2.2.9. КОМАНДЫ ДЕЛЕНИЯ

1. Разделить аккумулятор на 2 без учета знака:

```
ANA  A         ;ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА
RAR            ;РАЗДЕЛИТЬ НА 2, ОЧИСТИВ ЗНАК
```

2. Разделить аккумулятор на 4 без учета знака:

```
ANA  A         ;ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА
RAR            ;РАЗДЕЛИТЬ НА 2, ОЧИСТИВ ЗНАК
ANA  A         ;ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА И СНОВА
               ; РАЗДЕЛИТЬ НА 2
RAR
```

или

```
        RAR               ;ДВАЖДЫ СДВИНУТЬ ВПРАВО
        RAR
        ANI   00111111B   ;ЗАТЕМ ОЧИСТИТЬ ДВА СТАРШИХ РАЗРЯДА
```

В итоге, быстрее очистить в конце старшие разряды с помощью логического И, чем каждый раз очищать флаг переноса.

3. Разделить аккумулятор на 2 с учетом знака:

```
        RLC               ;СКОПИРОВАТЬ ЗНАКОВЫЙ РАЗРЯД В
                          ; РАЗРЯД 0 И ФЛАГ ПЕРЕНОСА
        RAR               ;РАЗДЕЛИТЬ НА 2, РАСШИРИВ ЗНАК
        RAR
```

Команда RLC создает две копии разряда 7: одну — во флаге переноса, а другую — в разряде 0 аккумулятора. Такая операция известна как *арифметический сдвиг*, так как при этом сохраняется знак числа, в то время как абсолютное значение уменьшается. Копирование знакового разряда в правые от него разряды называется *расширением знака.*

4. Разделить ячейки памяти ADDR и ADDR + 1 (старший байт в ADDR + 1) на 2 без учета знака:

```
        ANA   A           ;ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА
        LXI   H,ADDR+1
        MOV   A,M         ;ВЗЯТЬ СТАРШИЙ БАЙТ
        RAR               ;СДВИНУТЬ СТАРШИЙ БАЙТ ВПРАВО ЛОГИЧЕСКИ
        MOV   M,A
        DCX   H
        MOV   A,M         ;ПЕРЕСЛАТЬ ФЛАГ ПЕРЕНОСА В МЛАДШИЙ БАЙТ
        RAR
        MOV   M,A
```

5. Разделить ячейки памяти ADDR и ADDR + 1 (старший байт в ADDR + 1) на 2 с учетом знака:

```
        LXI   H,ADDR+1
        MOV   A,M         ;СДВИНУТЬ СТАРШИЙ БАЙТ ВПРАВО АРИФМЕТИЧЕСКИ
        RLC
        RAR
        RAR
        MOV   M,A
        DCX   H
        MOV   A,M         ;ПЕРЕСЛАТЬ ФЛАГ ПЕРЕНОСА В МЛАДШИЙ БАЙТ
        RAR
        MOV   M,A
```

6. Разделить пару регистров на 2 без учета знака:

```
        ANA   A           ;ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА
        MOV   A,RPH       ;СДВИНУТЬ СТАРШИЙ БАЙТ ВПРАВО ЛОГИЧЕСКИ
        RAR
        MOV   RPH,A
        MOV   A,RPL       ;ПЕРЕСЛАТЬ ФЛАГ ПЕРЕНОСА В МЛАДШИЙ БАЙТ
        RAR
        MOV   RPL,A
```

7. Разделить пару регистров на 2 с учетом знака:

```
        MOV   A,RPH       ;СДВИНУТЬ СТАРШИЙ БАЙТ ВПРАВО
        RLC               ; АРИФМЕТИЧЕСКИ
```

```
        RAR
        RAR
        MOV   RPH,A
        MOV   A,RPL         ;ПЕРЕСЛАТЬ ФЛАГ ПЕРЕНОСА В МЛАДШИЙ БАЙТ
        RAR
        MOV   RPL,A
```

2.2.10. КОМАНДЫ СРАВНЕНИЙ

1. Сравнить поразрядно VALUE с аккумулятором, установив в 1 несовпадающие разряды:

```
        XRI   VALUE
```

Напомним, что операция ИСКЛЮЧАЮЩЕЕ ИЛИ для двух разрядов равна 1, если и только если значения этих разрядов отличаются.

2. Сравнить поразрядно регистр с аккумулятором, установив в 1 несовпадающие разряды:

```
        XRA   REG
```

3. Сравнить пары регистров. Установить флаг переноса, если RP1 > RP2 (считается, что числа без знака) и очистить флаг переноса в противном случае. Установить флаг нуля, если две пары равны, и очистить флаг нуля в противном случае:

```
        MOV   A,RP2H        ;СРАВНИТЬ СТАРШИЕ БАЙТЫ
        CMP   RP1H          ;СТАРШИЕ БАЙТЫ РАВНЫ?
        JNZ   DONE          ;НЕТ, ЗАКОНЧИТЬ СРАВНЕНИЕ
        MOV   A,RP2L        ;ДА, СРАВНИТЬ МЛАДШИЕ БАЙТЫ
        CMP   RP1L          ;УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА, ЕСЛИ RP1 БОЛЬШЕ
DONE:   NOP
```

4. Сравнить пары регистров (RP1 и RP2). Установить флаг переноса, если RP1 > RP2 (считается, что числа без знака), и очистить флаг переноса в противном случае:

```
        MOV   A,RP2L        ;УСТАНОВИТЬ ЗАЕМ ПО МЛАДШИМ БАЙТАМ
        CMP   RP1L
        MOV   A,RP2H        ;ЗАТЕМ СРАВНИТЬ СТАРШИЕ БАЙТЫ С УЧЕТОМ ЗАЕМА
        SBB   RP1H
```

Чтобы учесть при сравнении заем из младших байтов, для старших байтов используется команда SBB. Флаг нуля отражает результат только второго вычитания, т. е. флаг нуля устанавливается в 1, если RP2 > RP1, но их разность меньше, чем 100_{16}.

5. Сравнить пару регистров с 16-разрядным числом. Установить флаг переноса, если 16-разрядное число меньше или равно паре регистров (считается, что числа без знака), и очистить флаг переноса в противном случае. Предполагается, что парой регистров являются не H и L:

```
        LXI   H,-VAL16  ;ВЗЯТЬ ОТРИЦАТЕЛЬНОЕ ЗНАЧЕНИЕ КОНСТАНТЫ
        DAD   RP
```

Для сравнения с регистрами H и L можно использовать

```
        LXI   D,-VAL16  ;ВЗЯТЬ ОТРИЦАТЕЛЬНОЕ ЗНАЧЕНИЕ КОНСТАНТЫ
        DAD   D
```

6. Сравнение блоков (так же, как в микропроцессоре Z 80). Сравнить аккумулятор с байтами в памяти, начиная с адреса, содержащегося в регистрах H и L. Продолжать до тех пор, пока не будет найден совпадающий байт (при этом флаг переноса равен 0), или пока счетчик (регистр B) не уменьшится до 0 (флаг переноса равен 1):

```
CMPBYT:   CMP  M        ;ПРОВЕРИТЬ ТЕКУЩИЙ БАЙТ
          JZ   DONE     ;ЕСЛИ ЕСТЬ СОВПАДЕНИЕ, ТО ПРОВЕРКА ОКОНЧЕНА
          INX  H        ;ИНАЧЕ, ПЕРЕЙТИ К СЛЕДУЮЩЕМУ БАЙТУ
          DCR  B
          JNZ  CMPBYT   ;ЕСЛИ ОСТАЛОСЬ ЧТО-ЛИБО ДЛЯ СРАВНЕНИЯ,
                        ; ТО ПЕРЕЙТИ НА НАЧАЛО ЦИКЛА
          STC           ;ЕСЛИ НЕТ, УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА
DONE:     NOP
```

Напомним, что сравнение двух равных чисел очищает флаг переноса.

2.2.11. КОМАНДЫ ПОЛУЧЕНИЯ ДОПОЛНЕНИЯ ДО ДВУХ (ОТРИЦАТЕЛЬНОГО ЧИСЛА)

1. Сделать аккумулятор отрицательным:

```
          CMA           ;ИНВЕРТИРОВАННЫЙ КОД
          INR  A        ;ДОПОЛНЕНИЕ ДО ДВУХ
```

Дополнение до двух есть дополнение до одного плюс 1.

2. Сделать регистр отрицательным:

```
          SUB  A        ;СФОРМИРОВАТЬ 0 - REG
          SUB  REG
          MOV  REG,A
```

3. Сделать ячейку памяти ADDR отрицательной:

```
          LXI  H,ADDR
          SUB  A        ;СФОРМИРОВАТЬ 0 - (ADDR)
          SUB  M
          MOV  M,A
```

4. Сделать пару регистров отрицательной:

```
          MOV  A,RPH    ;ИНВЕРТИРОВАННЫЙ КОД
          CMA
          MOV  RPH,A
          MOV  A,RPL
          CMA
          MOV  RPL,A
          INX  RP       ;ДОБАВИТЬ 1 ДЛЯ ДОПОЛНЕНИЯ ДО ДВУХ
```

5. Получить дополнение аккумулятора до 9 (т. е. заменить (A) на 99 – (A)):

```
          MOV  REG,A
          MOV  A,99H
          SUB  REG
```

Здесь нет необходимости в команде DAA, так как, если аккумулятор содержит правильное число в коде BCD, то и 99 – (A) всегда будет правильным числом в этом коде.

6. Получить дополнение аккумулятора до 10 (т. е. заменить (А) на 100 – (А)):

```
MOV  REG,A      ;СФОРМИРОВАТЬ ДОПОЛНЕНИЕ ДО ДЕВЯТИ
MVI  A,99H
SUB  REG
ADI  1          ;ЗАТЕМ ДОБАВИТЬ 1 В ДЕСЯТИЧНОМ ВИДЕ
DAA
```

Напомним, что команда DAA правильно выполняется только после команд сложения.

2.2.12. КОМАНДЫ РАСШИРЕНИЯ

1. Расширить аккумулятор до 16-разрядного числа без знака в паре регистров:

```
MOV  RPL,A      ;ПЕРЕСЛАТЬ 8 РАЗРЯДОВ
MVI  RPH,0      ;РАСШИРИТЬ 8 РАЗРЯДОВ ДО 16-ТИ
```

Эта процедура позволяет использовать значение, содержащееся в аккумуляторе, в качестве индекса. Команда DAD добавляет затем индекс к базе.

2. Расширить аккумулятор до 16-разрядного числа со знаком в паре регистров:

```
MOV  RPL,A      ;ПЕРЕСЛАТЬ 8 РАЗРЯДОВ
ADD  A          ;ПЕРЕСЛАТЬ ЗНАКОВЫЙ РАЗРЯД ВО ФЛАГ ПЕРЕНОСА
SBB  A          ;ВЫЧЕСТЬ ЗНАКОВЫЙ РАЗРЯД ИЗ НУЛЯ
MOV  RPH,A      ;РАСШИРИТЬ 8 РАЗРЯДОВ ДО 16-ТИ
```

В результате команды SBB A получается 00, если флаг переноса равен 0, и FF_{16}, если знак переноса равен 1. Таким образом, флаг переноса распространяется по всему аккумулятору.

3. Расширить содержимое ячейки памяти ADDR до 16-разрядного числа со знаком в ячейках памяти ADDR (младший байт) и ADDR + 1 (старший байт):

```
LXI  H,ADDR     ;ВЫБРАТЬ ЧИСЛО
MOV  A,M
ADD  A          ;ПЕРЕСЛАТЬ ЗНАКОВЫЙ РАЗРЯД
                ; ВО ФЛАГ ПЕРЕНОСА
SBB  A          ;СФОРМИРОВАТЬ ЗНАКОВЫЙ БАЙТ
INX  H          ;ЗАПОМНИТЬ ЗНАКОВЫЙ БАЙТ
MOV  M,A
```

4. Расширить разряд 0 аккумулятора по всему аккумулятору, т. е. получить (А) = 00, если разряд 0 = 0 и FF_{16}, если разряд 0 = 1:

```
RAR             ;ПЕРЕСЛАТЬ РАЗРЯД 0 ВО ФЛАГ ПЕРЕНОСА
SBB  A          ;СФОРМИРОВАТЬ 0 - РАЗРЯД 0
```

5. Функция Sign (знак). Заменить значение в аккумуляторе на 00, если оно положительное, и на FF_{16}, если оно отрицательное:

```
ADD  A          ;ПЕРЕСЛАТЬ ЗНАКОВЫЙ РАЗРЯД ВО ФЛАГ ПЕРЕНОСА
SBB  A          ;СФОРМИРОВАТЬ 0 - ЗНАКОВЫЙ РАЗРЯД
```

2.3. ЛОГИЧЕСКИЕ КОМАНДЫ

Эта группа включает следующие команды: логическое И, логическое ИЛИ, логическое ИСКЛЮЧАЮЩЕЕ ИЛИ, логическое НЕ (дополнение),

сдвиг, циклический сдвиг и проверку. Она включает также те арифметические команды (такие, как сложение аккумулятора с самим собой), которые выполняют логические функции.

2.3.1. КОМАНДЫ ОПЕРАЦИИ ЛОГИЧЕСКОЕ И

1. Очистить разряды аккумулятора:

```
        ANI   MASK        ;ОЧИСТИТЬ РАЗРЯД С ПОМОЩЬЮ МАСКИ
```

Константа MASK имеет нули в тех разрядах, которые должны быть очищены, и единицы в тех разрядах, которые должны остаться без изменения. Например:

```
        ANI   11011011B ;ОЧИСТИТЬ РАЗРЯДЫ 2 И 5
```

Напомним, что операция логическое И разряда с единицей оставляет его без изменения.

2. Проверить разряды — установить флаги так, как будто выполнена операция логическое И аккумулятора с регистром или ячейкой памяти, но при этом оставить аккумулятор без изменения:

```
        MOV   REG1,A      ;СОХРАНИТЬ АККУМУЛЯТОР
        ANA   REG         ;ВЫПОЛНИТЬ ЛОГИЧЕСКОЕ "И"
        MOV   A,REG1      ;ВОССТАНОВИТЬ АККУМУЛЯТОР
```

Эта последовательность команд основана на том, что MOV не влияет на флаги.

3. Проверить разряды аккумулятора:

```
        ANI   MASK        ;ПРОВЕРИТЬ РАЗРЯДЫ С ПОМОЩЬЮ МАСКИ
```

Константа MASK имеет единицы в тех разрядах, которые должны быть проверены, и нули в остальных разрядах. Флаг нуля устанавливается в 1, если все проверяемые разряды равны 0, и в 0 — в противном случае. Затем с помощью команды JZ или JNZ можно сделать переход в зависимости от содержания проверяемых разрядов.

Например:

```
        ANI   01000000B ;ПРОВЕРИТЬ РАЗРЯД 6
```

Результат равен 0, если разряд 6 аккумулятора равен 0, и 0100000_2, если разряд 6 равен 1. При этом флаг нуля содержит логическое дополнение разряда 6.

4. Логическое И непосредственного операнда с флагами (кодами условий), Выполнить операцию логическое И байта данных, заданного непосредственно, при этом очистить те флаги, для которых разряды в маске равны 0:

```
        PUSH PSW          ;ПЕРЕСЛАТЬ PSW В ПАРУ РЕГИСТРОВ
        POP   RP
        MVI   A,MASK      ;ОЧИСТИТЬ ФЛАГИ
        ANA   RPL
        MOV   RPL,A
        PUSH RP           ;ВОССТАНОВИТЬ PSW С ОЧИЩЕННЫМИ ФЛАГАМИ
        POP   PSW
```

Эта последовательность команд изменяет пару регистров (B, D или H).

2.3.2. КОМАНДЫ ОПЕРАЦИИ ЛОГИЧЕСКОЕ ИЛИ

1. Установить разряды аккумулятора:

```
        ORI   MASK        ;УСТАНОВИТЬ РАЗРЯДЫ С ПОМОЩЬЮ МАСКИ
```

Константа MASK имеет единицы в тех разрядах, которые должны быть установлены, и нули в остальных. Например:

```
ORI  00010010B ;УСТАНОВИТЬ РАЗРЯДЫ 1 И 4
```

Напомним, что операция логическое ИЛИ разряда с нулем оставляет разряд без изменения.

2. Проверить пару регистров на 0. Установить флаг нуля, если оба регистра пары равны 0:

```
MOV  A,RPH      ;ПРОВЕРИТЬ ПАРУ РЕГИСТРОВ НА НУЛЬ
ORA  RPL
```

Флаг нуля устанавливается, если и только если оба байта пары регистров равны 0. При этом изменяются также аккумулятор и остальные флаги.

3. Логическое ИЛИ непосредственного операнда с флагами (кодами условий). Выполнить операцию логическое ИЛИ байта данных, заданного непосредственно, с регистром флагов, при этом установить те флаги, для которых разряды в маске равны 1:

```
PUSH PSW        ;ПЕРЕСЛАТЬ PSW В ПАРУ РЕГИСТРОВ
POP  RP
MVI  A,MASK     ;ОЧИСТИТЬ ФЛАГИ
ORA  RPL
MOV  RPL,A
PUSH RP         ;ВОССТАНОВИТЬ PSW С ОЧИЩЕННЫМИ ФЛАГАМИ
POP  PSW
```

Эта последовательность команд изменяет пару регистров (B, D или H).

2.3.3. КОМАНДЫ ОПЕРАЦИИ ИСКЛЮЧАЮЩЕЕ ИЛИ

1. Инвертировать разряды аккумулятора:

```
XRI  MASK       ;ИНВЕРТИРОВАТЬ РАЗРЯДЫ С ПОМОЩЬЮ МАСКИ
```

Константа MASK имеет единицы в тех разрядах, которые должны быть инвертированы, и нули в разрядах, которые должны остаться без изменения. Например:

```
XRI  11000000B ;ИНВЕРТИРОВАТЬ РАЗРЯДЫ 6 И 7
```

Напомним, что логическая операция ИСКЛЮЧАЮЩЕЕ ИЛИ разряда с нулем оставляет разряд без изменения.

2. Инвертировать аккумулятор, установив при этом флаги:

```
XRI  11111111B ;ИНВЕРТИРОВАТЬ А И УСТАНОВИТЬ ФЛАГИ
```

Логическая операция ИСКЛЮЧАЮЩЕЕ ИЛИ со всеми единицами инвертирует все разряды. Эта команда отличается от CMA только тем, что она влияет на все флаги, в то время как CMA не изменяет ни один флаг.

3. Сравнить поразрядно регистр с аккумулятором, установив каждый отличающийся разряд:

```
XRA  REG        ;ПОРАЗРЯДНОЕ СРАВНЕНИЕ
```

Операция ИСКЛЮЧАЮЩЕЕ ИЛИ выполняет ту же функцию, что и операция "не равно". Заметим, что флаг знака равен 1, если значения обоих операндов в разряде 7 отличаются.

4. Логически добавить регистр к аккумулятору (т. е. без переносов между разрядами):

```
XRA  REG        ;ЛОГИЧЕСКОЕ СЛОЖЕНИЕ
```

Операция ИСКЛЮЧАЮЩЕЕ ИЛИ выполняет ту же функцию, что и операция поразрядного сложения без переносов. Логические суммы часто используются для получения контрольных сумм и кодов определения и исправления ошибки.

2.3.4. КОМАНДЫ ОПЕРАЦИИ ЛОГИЧЕСКОЕ НЕ.

1. Инвертировать аккумулятор, установив при этом флаги:

```
XRI  11111111B ;ИНВЕРТИРОВАТЬ А И УСТАНОВИТЬ ФЛАГИ
```

Логическое ИСКЛЮЧАЮЩЕЕ ИЛИ со всеми единицами инвертирует все разряды. Эта команда отличается от СМА только тем, что влияет на флаги, в то время как СМА не влияет.

2. Инвертировать какие-либо разряды аккумулятора:

```
XRI  MASK       ;ИНВЕРТИРОВАТЬ РАЗРЯДЫ С ПОМОЩЬЮ МАСКИ
```

Константа MASK имеет единицы в тех разрядах, которые должны быть инвертированы, и нули в разрядах, которые должны остаться без изменения. Например:

```
XRI  01010001B ;ИНВЕРТИРОВАТЬ РАЗРЯДЫ 0,4 И 6
```

Напомним, что логическая операция ИСКЛЮЧАЮЩЕЕ ИЛИ разряда с нулем оставляет разряд без изменения.

3. Инвертировать ячейку памяти ADDR:

```
LXI  H,ADDR
MOV  A,M        ;ПОЛУЧИТЬ ДАННЫЕ
CMA             ;ИНВЕРТИРОВАТЬ
MOV  M,A        ;ЗАПОМНИТЬ РЕЗУЛЬТАТ
```

Команда СМА применима только к аккумулятору.

4. Инвертировать разряд 0 регистра:

```
INR  REG
```

или

```
DCR  REG
```

Каждая из этих команд может изменить и другие разряды регистра. Результирующее значение разряда 0 будет равно 0, если вначале оно равнялось 1, и 1, если его начальное значение было 0.

5. Инвертировать разряд 0 ячейки памяти:

```
LXI  H,ADDR
INR  M
```

или

```
LXI  H,ADDR
DCR  M
```

6. Инвертировать цифру в аккумуляторе.

• Младшую цифру:

```
XRI  00001111B ;ИНВЕРТИРОВАТЬ МЛАДШУЮ ЦИФРУ
```

• Старшую цифру:

```
XRI  11110000B ;ИНВЕРТИРОВАТЬ СТАРШУЮ ЦИФРУ
```

Приведенные процедуры полезны в том случае, когда аккумулятор содержит десятичное число при отрицательной логике, как, например, при вводе с обычного десятипозиционного наборного диска или пакетного переключателя.

2.3.5. КОМАНДЫ СДВИГА

1. Сдвинуть аккумулятор вправо логически:

```
ANA  A          ;ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА
RAR             ;СДВИНУТЬ ВПРАВО ЛОГИЧЕСКИ
```

Команда ANA A (или ORA A) очищает флаг переноса без изменения аккумулятора. Альтернативой является

```
RAR             ;СДВИНУТЬ ВПРАВО ЦИКЛИЧЕСКИ
```

2. Сдвинуть аккумулятор вправо арифметически, сохранив при этом знаковый разряд:

```
RLC             ;СКОПИРОВАТЬ ЗНАКОВЫЙ РАЗРЯД
                ; В РАЗРЯД 0 И ФЛАГ ПЕРЕНОСА
RAR             ;ЗАТЕМ ДВАЖДЫ СДВИНУТЬ НАЗАД
RAR
```

Команда RLC записывает старое значение разряда 7 как во флаг переноса, так и в разряд 0 аккумулятора.

3. Сдвинуть аккумулятор влево логически:

```
ADD  A          ;СДВИНУТЬ ВЛЕВО ЛОГИЧЕСКИ
```

Сложение аккумулятора с самим собой эквивалентно логическому сдвигу влево.

4. Сдвинуть регистры H и L влево логически:

```
DAD  H          ;СДВИНУТЬ HL ВЛЕВО ЛОГИЧЕСКИ
```

Сложение регистров H и L самих с собой эквивалентно 16-разрядному логическому сдвигу влево.

5. Сдвинуть пару регистров вправо логически:

```
ANA  A          ;ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА
MOV  A,RPH      ;СДВИНУТЬ СТАРШИЙ БАЙТ ВПРАВО ЛОГИЧЕСКИ
RAR
MOV  RPH,A
MOV  A,RPL      ;СВИНУТЬ МЛАДШИЙ БАЙТ ВПРАВО ЦИКЛИЧЕСКИ,
                ; ПРИХВАТИВ ФЛАГ ПЕРЕНОСА
RAR
MOV  RPL,A
```

Основным моментом здесь является то, что младший байт должен быть сдвинут циклически вместе с флагом переноса, являющимся результатом логического сдвига старшего байта.

Сдвинуть пару регистров вправо арифметически:

```
        MOV   A,RPH      ;СДВИНУТЬ СТАРШИЙ БАЙТ ВПРАВО АРИФМЕТИЧЕСКИ
        RLC
        RAR
        RAR
        MOV   RPH,A
        MOV   A,RPL      ;СДВИНУТЬ МЛАДШИЙ БАЙТ ВПРАВО ЦИКЛИЧЕСКИ
        RAR
        MOV   RPL,A
```

Здесь сдвиг младшего байта сходен с логическим сдвигом.

7. Поменять местами цифры в аккумуляторе, т. е. заменить четыре младших разряда на четыре старших разряда, и наоборот:

```
        RLC              ;СДВИГ ЦИФРЫ = 4 ЦИКЛИЧЕСКИХ СДВИГА
        RLC
        RLC
        RLC
```

или

```
        RRC              ;СДВИГ ЦИФРЫ = 4 ЦИКЛИЧЕСКИХ СДВИГА
        RRC
        RRC
        RRC
```

8. Нормализовать аккумулятор, т. е. сдвигать его содержимое влево до тех пор, пока его старший разряд не будет равен 1. Не сдвигать совсем, если аккумулятор содержит 0:

```
        ANA   A          ;ПРОВЕРИТЬ АККУМУЛЯТОР
        JM    DONE       ;ВЫЙТИ,ЕСЛИ УЖЕ НОРМАЛИЗОВАН
        JZ    DONE       ;ВЫЙТИ, ЕСЛИ НУЛЬ
SHIFT:  ADD   A          ;ИНАЧЕ, СДВИНУТЬ А ВЛЕВО НА 1 РАЗРЯД
        JP    SHIFT      ;СДВИГАТЬ, ПОКА НЕ БУДЕТ НОРМАЛИЗОВАН
DONE:   NOP
```

2.3.6. КОМАНДЫ ЦИКЛИЧЕСКОГО СДВИГА

1. Циклически сдвинуть пару регистров вправо:

```
        MOV   A,RPL      ;ПЕРЕСЛАТЬ РАЗРЯД 0 ВО ФЛАГ ПЕРЕНОСА
        RAR
        MOV   A,RPH      ;СДВИНУТЬ СТАРШИЙ БАЙТ ВПРАВО ЦИКЛИЧЕСКИ
        RAR
        MOV   RPH,A
        MOV   A,RPL      ;СДВИНУТЬ МЛАДШИЙ БАЙТ ВПРАВО ЦИКЛИЧЕСКИ
        RAR
        MOV   RPL,A
```

Первые две команды пересылают разряд 0 пары регистров во флаг переноса.

2. Циклически сдвинуть пару регистров влево:

```
        MOV   A,RPH      ;ПЕРЕСЛАТЬ РАЗРЯД 15 ВО ФЛАГ ПЕРЕНОСА
        RAL
        MOV   A,RPL      ;СДВИНУТЬ МЛАДШИЙ БАЙТ ВЛЕВО ЦИКЛИЧЕСКИ
        RAL
        MOV   RPL,A
        MOV   A,RPH      ;СДВИНУТЬ СТАРШИЙ БАЙТ ВЛЕВО ЦИКЛИЧЕСКИ
        RAL
        MOV   RPH,A
```

Первые две команды пересылают разряд 15 пары регистров во флаг переноса.

3. Циклически сдвинуть аккумулятор влево через флаг переноса, установив флаги:

```
        ADC   A            ;СДВИНУТЬ ВЛЕВО ЦИКЛИЧЕСКИ,
                           ; УСТАНОВИВ ФЛАГИ
```

Эта команда сходна с RAL, за исключением того, что влияет на все флаги, в то время как RAL влияет только на флаг переноса.

4. Циклически сдвинуть пару регистров вправо через флаг переноса:

```
        MOV   A,RPH        ;СДВИНУТЬ СТАРШИЙ БАЙТ ВПРАВО ЦИКЛИЧЕСКИ
        RAR                ; ЧЕРЕЗ ФЛАГ ПЕРЕНОСА
        MOV   RPH,A
        MOV   A,RPL        ;СДВИНУТЬ МЛАДШИЙ БАЙТ ВПРАВО ЦИКЛИЧЕСКИ
        RAR
        MOV   RPL,A
```

5. Циклически сдвинуть пару регистров влево через флаг переноса:

```
        MOV   A,RPL        ;СДВИНУТЬ МЛАДШИЙ БАЙТ ВЛЕВО
        RAL                ; ЧЕРЕЗ ФЛАГ ПЕРЕНОСА
        MOV   RPL,A
        MOV   A,RPH        ;СДВИНУТЬ СТАРШИЙ БАЙТ ВЛЕВО ЦИКЛИЧЕСКИ
        RAL
        MOV   RPH,A
```

2.3.7. КОМАНДЫ ПРОВЕРКИ

1. **Проверить аккумулятор.** Установить флаги в соответствии со значением аккумулятора без изменения самого значения:

```
        ANA   A            ;ПРОВЕРИТЬ АККУМУЛЯТОР
```

или

```
        ORA   A            ;ПРОВЕРИТЬ АККУМУЛЯТОР
```

В обоих случаях очищается флаг переноса.

2. **Проверить регистр.** Установить флаги в соответствии со значением регистра без изменения самого значения:

```
        INR   REG          ;ПРОВЕРИТЬ РЕГИСТР
        DCR   REG
```

Эта последовательность команд не влияет на флаг переноса или аккумулятор.

3. **Проверить ячейку памяти.** Установить флаги в соответствии со значением ячейки памяти, не изменяя самого значения:

```
        LXI   H,ADDR       ;ПРОВЕРИТЬ ЯЧЕЙКУ ПАМЯТИ ADDR
        INR   M
        DCR   M
```

Эта последовательность команд не влияет на флаг переноса или аккумулятор.

4. **Проверить пару регистров.** Установить флаг нуля в соответствии со значением пары регистров без изменения этого значения:

```
        MOV   A,RPH        ;ПРОВЕРИТЬ ПАРУ РЕГИСТРОВ
        ORA   RPL
```

Эта последовательность команд изменяет аккумулятор и остальные флаги.

5. **Проверить разряды аккумулятора.** Установить флаг нуля, если все проверяемые разряды равны 0, и очистить флаг нуля в противном случае:

```
ANI  MASK        ;ПРОВЕРИТЬ РАЗРЯДЫ С ПОМОЩЬЮ МАСКИ
```

Константа MASK содержит единицы в проверяемых разрядах и нули во всех остальных. Флаг нуля устанавливается в 1, если все проверяемые разряды содержат нули, и в 0 – в противном случае. Флаг нуля может быть затем использован в качестве условия перехода. Например:

```
ANI  00001000B ;ПРОВЕРИТЬ РАЗРЯД 3
```

Результат равен 0, если разряд 3 аккумулятора равен 0, ии 00001000_2, если этот разряд равен 1. В результате флаг нуля содержит логический обратный код разряда 3.

6. **Сравнить поразрядно регистр с аккумулятором.** Установить в 1 каждый несовпадающий разряд:

```
XRA  REG         ;ПОРАЗРЯДНОЕ СРАВНЕНИЕ
```

Операция ИСКЛЮЧАЮЩЕЕ ИЛИ выполняет ту же функцию, что и операция "не равно".

7. **Проверка разряда.** Установить флаги так же, как при операции логическое И между аккумулятором и регистром или ячейкой памяти, но без изменения аккумулятора:

```
MOV  REG1,A    ;СОХРАНИТЬ АККУМУЛЯТОР
ANA  REG       ;ВЫПОЛНИТЬ ЛОГИЧЕСКОЕ "И"
MOV  A,REG1    ;ВОССТАНОВИТЬ АККУМУЛЯТОР
```

2.4. КОМАНДЫ ПЕРЕДАЧИ ДАННЫХ

Эта группа включает команды загрузки, запоминания, пересылки, обмена, ввода, вывода, очистки и установки. Кроме того, она включает арифметические команды (такие как вычитание аккумулятора из самого себя), которые заносят определенное значение или содержимое какого-либо регистра в аккумулятор или другой регистр назначения, не изменяя при этом данных.

2.4.1. КОМАНДЫ ЗАГРУЗКИ

1. Загрузить регистр прямо:

```
LDA  ADDR
MOV  REG,A
```

или

```
LXI  H,ADDR
MOV  REG,M
```

В первом случае используется аккумулятор, в то время как во втором – регистры H и L.

2. Загрузить аккумулятор косвенно.

• Из ячейки памяти, адрес которой содержится в паре регистров HL:

```
MOV  A,M
```

• Из ячейки памяти, адрес которой содержится в паре регистров BC или DE:

```
LDAX RP
```

3. Загрузить регистр косвенно.

• Из ячейки памяти, адрес которой содержится в паре регистров HL:

```
MOV  REG,M
```

• Из ячейки памяти, адрес которой содержится в паре регистров BC или DE:

```
LDAX RP
MOV  REG,A
```

4. Загрузить пару регистров прямо.

• H и L:

```
LHLD ADDR
```

• D и E:

```
LHLD ADDR
XCHG
```

• B и C

```
LHLD ADDR
MOV  B,H
MOV  C,L
```

5. Загрузить указатель стека прямо:

```
LHLD ADDR
SPHL
```

6. Загрузить регистры H и L косвенно из ячеек памяти, адрес которых содержится в поле регистров HL:

```
MOV  A,M        ;ЗАГРУЗИТЬ МЛАДШИЙ БАЙТ
INX  H
MOV  H,M        ;ЗАГРУЗИТЬ СТАРШИЙ БАЙТ
MOV  L,A
```

7. Загрузить пару регистров (B или D) косвенно из ячеек памяти, адрес которых содержится в паре регистров HL:

```
MOV  RPL,M      ;ЗАГРУЗИТЬ МЛАДШИЙ БАЙТ
INX  H
MOV  RPH,M      ;ЗАГРУЗИТЬ СТАРШИЙ БАЙТ
DCX  H          ;ВОССТАНОВИТЬ НАЧАЛЬНОЕ ЗНАЧЕНИЕ HL
```

8. Загрузить регистр флагов прямо:

```
LHLD ADDR       ;ЗАГРУЗИТЬ L ИЗ ADDR
PUSH H          ;HL В СТЕК, L В ВЕРШИНУ
POP  PSW        ;HL В PSW, ПРИ ЭТОМ L ВО ФЛАГИ
```

Эта процедура позволяет пользователю при отладке или проверке задать начальное значение регистра флагов. Заметим, что она изменяет аккумулятор и младшую по значению половину пары регистров.

9. Загрузить в регистр флагов 8-разрядное число VALUE:

```
MVI  RPL,VALUE ;ЗАПИСАТЬ VALUE В МЛАДШИЙ БАЙТ
               ; ПАРЫ РЕГИСТРОВ
```

```
PUSH RP          ;ПЕРЕСЛАТЬ ВО ФЛАГИ ЧЕРЕЗ СТЕК
POP  PSW
```

Выполнение этой последовательности команд осложняется тем, что загрузить в стек и получить из стека можно содержимое только пары регистров.

10. Загрузить маски прерываний (регистр I) прямо (только для 8085):

```
LDA  ADDR        ;ВЗЯТЬ ЗНАЧЕНИЕ
SIM              ;УСТАНОВИТЬ МАСКИ ПРЕРЫВАНИЙ
```

11. Загрузить в маски прерываний (регистр I) 8-разрядное значение VALUE (только в 8085):

```
MVI  A,VALUE     ;ВЗЯТЬ ЗНАЧЕНИЕ
SIM              ;УСТАНОВИТЬ МАСКИ ПРЕРЫВАНИЙ
```

12. Загрузить в ячейки памяти PTR и PTR + 1 значение ADDR (старший байт – в PTR + 1):

```
LXI  H,ADDR      ;ВЗЯТЬ КОСВЕННЫЙ АДРЕС
SHLD PTR         ;СОХРАНИТЬ КОСВЕННЫЙ АДРЕС В ПАМЯТИ
```

2.4.2. КОМАНДЫ ЗАПОМИНАНИЯ

1. Запомнить регистр прямо:

```
MOV  A,REG
STA  ADDR
```

или

```
LXI  H,ADDR
MOV  M,REG
```

В первом случае используется аккумулятор, в то время как во втором – регистры H и L.

2. Запомнить аккумулятор косвенно.

- По адресу, содержащемуся в HL:

```
MOV  M,A
```

- По адресу, содержащемуся в паре регистров BC или DE:

```
STAX RP
```

3. Запомнить регистр косвенно.

- По адресу, содержащемуся в паре регистров HL:

```
MOV  M,REG
```

- По адресу, содержащемуся в паре регистров DE или BC:

```
MOV  A,REG
STAX RP
```

4. Запомнить пару регистров прямо.

- H и L:

```
SHLD ADDR
```

- D и E:

```
XCHG
SHLD ADDR
```

• B и C:

```
MOV  H,B
MOV  L,C
SHLD ADDR
```

5. Запомнить указатель стека прямо:

```
LXI  H,0
DAD  SP
SHLD ADDR
```

Единственный способ определить значение указателя стека – это добавить его к H и L.

6. Запомнить пару регистров (B или D) косвенно по адресу в паре регистров HL:

```
MOV  M,RPL      ;ЗАПОМНИТЬ МЛАДШИЙ БАЙТ
INX  H
MOV  M,RPH      ;ЗАПОМНИТЬ СТАРШИЙ БАЙТ
DCX  H          ;ВОССТАНОВИТЬ НАЧАЛЬНОЕ ЗНАЧЕНИЕ HL
```

Пара регистров запоминается в памяти в обычном перевернутом формате.

7. Запомнить регистр флагов прямо:

```
PUSH PSW        ;F В ВЕРШИНУ СТЕКА
POP  H          ;F В L
SHLD ADDR       ;F В ADDR, ADDR + 1 ТЕРЯЕТСЯ
```

или

```
PUSH PSW        ;F В ВЕРШИНУ СТЕКА
POP  H          ;F В L
MOV  A,L        ;F В A
STA  ADDR       ;F В ADDR
```

8. Запомнить маски прерываний (регистр I) прямо (только в 8085):

```
RIM             ;ВЗЯТЬ МАСКИ ПРЕРЫВАНИЙ
STA  ADDR       ;ЗАПОМНИТЬ РЕГИСТР I В ПАМЯТИ
```

Заметим, что регистр I при чтении с помощью команды RIM отличается от регистра I при записи командой SIM.

2.4.3. КОМАНДЫ ПЕРЕСЫЛКИ

1. Переслать аккумулятор в регистр флагов:

```
MOV  RPL,A
PUSH RP
POP  PSW
```

Регистр флагов является младшим байтом слова состояния процессора (PSW). Приведенная последовательность команд изменяет также аккумулятор и младшую половину пары регистров. Аккумулятор можно сохранить, добавив в начало этой программы команду MOV RPH, A.

2. Переслать регистр флагов в аккумулятор:

```
PUSH PSW
POP  RP
MOV  A,RPL
```

Эта последовательность команд изменяет обе половины пары регистров RP.

3. Переслать пару регистров 1 в пару регистров 2:

```
MOV  RP2L,RP1L
MOV  RP2H,RP1H
```

Эта последовательность команд передает содержимое пары регистров RP1 в RP2 без изменения RP1. Напомним, что команда XCHG, в частности, обменивает пары регистров D и H.

4. Переслать пару регистров HL в указатель стека:

```
SPHL
```

5. Переслать указатель стека в пару регистров HL:

```
LXI  H,0
DAD  SP
```

6. Переслать содержимое пары HL в счетчик команд:

```
PCHL
```

7. Переслать блок (как в микропроцессоре Z80 с 16-разрядным счетчиком в паре регистров BC) [13, 14]. Передать данные из памяти, начиная с адреса, содержащегося в регистрах H и L, в память, начальный адрес которой содержится в регистрах D и E. Число переданных байтов содержится в регистрах B и C:

```
MOVBYT:  MOV  A,M      ;ВЗЯТЬ БАЙТ ДАННЫХ
         STAX D        ;И ПЕРЕСЛАТЬ ЕГО
         INX  D        ;УВЕЛИЧИТЬ УКАЗАТЕЛИ В БУФЕРАХ
         INX  H
         DCX  B        ;СОСЧИТАТЬ БАЙТЫ
         MOV  A,B
         ORA  C
         JNZ  MOVBYT
```

8. Переслать многократно (заполнить). Поместить содержимое аккумулятора в последовательные ячейки памяти, начиная с адреса, содержащегося в регистрах H и L. Число байтов, которые должны быть заполнены (один или больше), содержится в регистре B:

```
FILBYT:  MOV  M,A      ;ЗАПОЛНИТЬ ЯЧЕЙКУ ПАМЯТИ
         INX  H        ;УСТАНОВИТЬ АДРЕС СЛЕДУЮЩЕЙ ЯЧЕЙКИ
         DCR  B        ;СОСЧИТАТЬ БАЙТЫ
         JNZ  FILBYT
```

Эта последовательность команд может задавать начальное значение какого-либо массива или буфера. Если необходимо заполнить больше, чем 256 байт, то процедура усложняется. Если регистр E использовать для временного хранения, а число байтов, которые должны быть заполнены, находится в регистрах B и C, то эта программа будет иметь следующий вид:

```
         MOV  E,A      ;СОХРАНИТЬ ЗАПОЛНЯЮЩЕЕ ЗНАЧЕНИЕ
FILBYT:  MOV  M,E      ;ЗАПОЛНИТЬ ЯЧЕЙКУ ПАМЯТИ
         INX  X        ;УСТАНОВИТЬ АДРЕС СЛЕДУЮЩЕЙ ЯЧЕЙКИ
         DCX  B        ;СОСЧИТАТЬ БАЙТЫ, СОХРАНЯЯ ДАННЫЕ
         MOV  A,B
         ORA  C
         JNZ  FILBYT
```

Напомним, что команда MOV не влияет на флаги.

2.4.4. КОМАНДЫ ОБМЕНА

1. Обменять регистры, используя аккумулятор:

```
MOV  A,REG1
MOV  REG1,REG2
MOV  REG2,A
```

2. Обменять пары регистров.
- DE с HL:

```
XCHG
```

- BC с HL:

```
PUSH B          ;BC В ВЕРШИНУ СТЕКА
XTHL            ;BC В HL, HL В ВЕРШИНУ СТЕКА
POP  B          ;HL В BC
```

Команда XTHL обменивает пару регистров HL с вершиной стека.
- В общем случае RP1 с RP2

```
PUSH RP1        ;ПЕРЕСЛАТЬ RP1, RP2 В СТЕК
PUSH RP2
POP  RP1        ;ОБМЕНЯТЬ, ЧИТАЯ ИЗ СТЕКА
POP  RP2        ; В НЕПРАВИЛЬНОМ ПОРЯДКЕ
```

3. Обменять указатель стека с HL:

```
XCHG            ;HL В DE
LXI  H,0        ;SP В HL
DAD  SP
XCHG            ;SP В DE, ВОССТАНОВИТЬ HL
SPHL            ;HL В SP
XCHG            ;SP В HL
```

Эта процедура может быть использована для того, чтобы различить стек пользователя и стек операционной системы или монитора.

2.4.5. КОМАНДЫ ВВОДА

1. Поместить данные последовательного ввода (SID) во флаг переноса (только в 8085):

```
RIM             ;ВЫБРАТЬ ПОСЛЕДОВАТЕЛЬНЫЙ ВВОД
RAL             ;ПЕРЕСЛАТЬ SID ВО ФЛАГ ПЕРЕНОСА
```

Команда RIM помещает данные последовательного ввода в разряд 7 аккумулятора.

2. Ввести блок (как в микропроцессоре Z 80). Переслать блок данных из порта ввода IPORT в память, начиная с адреса, содержащегося в регистрах H и L.

- 8-разрядный счетчик байтов находится в регистре B:

```
BLKIN:  IN   IPORT    ;ПРОЧИТАТЬ БАЙТ
        MOV  M,A      ;СОХРАНИТЬ ВВЕДЕННЫЙ БАЙТ В ПАМЯТИ
        INX  H        ;ПОЛУЧИТЬ АДРЕС СЛЕДУЮЩЕЙ ЯЧЕЙКИ ПАМЯТИ
        DCR  B        ;СОСЧИТАТЬ БАЙТЫ
        JNZ  BLKIN
```

• 16-разрядный счетчик байтов находится в регистрах B и C:

```
BLKIN:    IN    IPORT     ;ПРОЧИТАТЬ БАЙТ
          MOV   M,A       ;СОХРАНИТЬ ВВЕДЕННЫЙ БАЙТ В ПАМЯТИ
          INX   H         ;ПОЛУЧИТЬ АДРЕС СЛЕДУЮЩЕЙ ЯЧЕЙКИ ПАМЯТИ
          DCX   B         ;СОСЧИТАТЬ БАЙТЫ
          MOV   A,B       ;ПРОВЕРИТЬ СЧЕТЧИК БАЙТОВ НА НУЛЬ
          ORA   C
          JNZ   BLKIN
```

Так как команда DCX не влияет ни на какие флаги, то пару регистров необходимо проверять на нуль.

2.4.6. КОМАНДЫ ВЫВОДА

1. Выполнить последовательный ввод через линию SOD (только в 8085).

Для того чтобы изменить линию SOD (последовательный вывод данных), следует установить разряд разрешения последовательного вывода (разряд 6 регистра I). Необходимо также очистить разряд разрешения маски (разряд 3 регистра I), чтобы предотвратить изменение масок прерываний.

• Послать флаг переноса в качестве данных последовательного вывода (SOD):

```
     MVI   A,10000000B     ;УСТАНОВИТЬ ПЕРЕД СДВИГОМ РАЗРЯД 7
     RAR                   ;ПЕРЕСЛАТЬ ФЛАГ ПЕРЕНОСА В SOD,
                           ; СДЕЛАТЬ SOE = 1
     SIM                   ;ПОСЛАТЬ ФЛАГ ПЕРЕНОСА В ЛИНИЮ SOD
```

• Сделать SOD = 1:

```
     MVI   A,11000000B     ;УСТАНОВИТЬ SOD, SOE
     SIM                   ;СДЕЛАТЬ SOD = 1
```

• Сделать SOD = 0:

```
     MVI   A,01000000B     ;ОЧИСТИТЬ SOD, УСТАНОВИТЬ SOE
     SIM                   ;СДЕЛАТЬ SOD = 0
```

2. Вывести блок (как в микропроцессоре Z80). Переслать блок данных из памяти (начиная с адреса, содержащегося в регистрах H и L) в порт вывода OPORT.

• 8-разрядный счетчик находится в регистре B:

```
BLKOUT:   MOV   A,M       ;ВЫБРАТЬ БАЙТ ИЗ ПАМЯТИ
          OUT   OPORT     ;ПОСЛАТЬ БАЙТ В ПОРТ ВЫВОДА
          INX   H         ;ПОЛУЧИТЬ АДРЕС СЛЕДУЮЩЕЙ ЯЧЕЙКИ ПАМЯТИ
          DCR   B         ;СОСЧИТАТЬ БАЙТЫ
          JNZ   BLKOUT
```

• 16-разрядный счетчик находится в регистрах B и C:

```
BLKOUT:   MOV   A,M       ;ВЫБРАТЬ БАЙТ ИЗ ПАМЯТИ
          OUT   OPORT     ;ПОСЛАТЬ БАЙТ В ПОРТ ВЫВОДА
          INX   H         ;ПОЛУЧИТЬ АДРЕС СЛЕДУЮЩЕЙ ЯЧЕЙКИ ПАМЯТИ
          DCX   B         ;СОСЧИТАТЬ БАЙТЫ
          MOV   A,B       ;ПРОВЕРИТЬ 16-РАЗРЯДНЫЙ СЧЕТЧИК НА НУЛЬ
          ORA   C
          JNZ   BLKOUT
```

2.4.7. КОМАНДЫ ОЧИСТКИ

1. Очистить аккумулятор:

```
SUB  A
```

или

```
XRA  A
```

или

```
MVI  A,0
```

Третий вариант команды выполняется медленнее и занимает больше памяти, чем два остальных, но он не изменяет флаги.

2. Очистить регистр:

```
MVI  REG,0
```

3. Очистить ячейку памяти:

```
SUB  A
STA  ADDR
```

или

```
LXI  H,ADDR
MVI  M,0
```

Второй вариант выполняется медленнее первого, но не влияет на аккумулятор и флаги. Само собой разумеется, при этом варианте используются регистры H и I.

4. Очистить пару регистров:

```
LXI  RP,0
```

5. Очистить ячейки памяти ADDR и ADDR + 1:

```
LXI  H,0
SHLD ADDR
```

6. Очистить флаг переноса:

```
ANA  A
```

или

```
ORA  A
```

Все логические команды, за исключением CMA, очищают флаг переноса, но две приведенные здесь команды удобны тем, что не изменяют аккумулятор. Напомним, что операции И и ИЛИ разряда с самим собой не изменяют его значения. Для того чтобы очистить флаг переноса, не влияя на остальные флаги, используйте

```
STC             ;СНАЧАЛА УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА
CMC             ;ЗАТЕМ ОЧИСТИТЬ ЕГО С ПОМОЩЬЮ ИНВЕРТИРОВАНИЯ
```

Этот подход полезен в том случае, когда флаг переноса указывает на появление ошибки или отказа.

7. Очистить разряды аккумулятора:

```
ANI  MASK       ;ОЧИСТИТЬ РАЗРЯДЫ С ПОМОЩЬЮ МАСКИ
```

Константа MASK имеет нули в тех разрядах, которые должны быть очищены, и единицы в разрядах, которые должны остаться без изменения. Например:

```
ANI  10111110B ;ОЧИСТИТЬ РАЗРЯДЫ 0 И 6
```

Операция И разряда с 1 оставляет этот разряд без изменения.

2.4.8. КОМАНДЫ УСТАНОВКИ

1. Установить значение аккумулятора FF_{16} (в двоичном виде — все единицы):

```
MVI   A,0FFH
```

или

```
SUB   A
DCR   A
```

2. Установить значение регистра FF_{16}:

```
MVI   REG,0FFH
```

3. Установить значение ячейки памяти FF_{16}:

```
MVI   A,0FFH
STA   ADDR
```

или

```
LXI   H,ADDR
MVI   M,0FFH
```

4. Установить разряды аккумулятора:

```
ORI   MASK       ;УСТАНОВИТЬ РАЗРЯДЫ С ПОМОЩЬЮ МАСКИ
```

Константа MASK имеет единицы в тех разрядах, которые должны быть установлены, и нули в остальных. Например:

```
ORI   10000000B ;УСТАНОВИТЬ РАЗРЯД 7 (ЗНАКОВЫЙ РАЗРЯД)
```

Операция ИЛИ разряда с нулем оставляет разряд без изменения.

2.5. КОМАНДЫ ПЕРЕХОДА

2.5.1. КОМАНДЫ БЕЗУСЛОВНОГО ПЕРЕХОДА

1. Перейти косвенно.

- По адресу в регистрах H и L:

```
PCHL
```

- По адресу в вершине стека:

```
RET
```

Заметим, что RET является обычной командой косвенного перехода, которая получает адрес назначения из вершины стека. Эту команду можно использовать не только для возврата из подпрограммы, но и для других целей.

- По адресу в регистрах D и E:

```
XCHG
PCHL
```

- По адресу в регистрах B и C:

```
MOV   H,B
MOV   L,C
PCHL
```

или

```
PUSH  B
RET
```

Второй вариант выполняется значительное медленнее, чем первый, но не изменяет HL.

• По адресу в ячейках памяти ADDR и ADDR + 1:

```
         LHLD ADDR       ;ВЫБРАТЬ КОСВЕННЫЙ АДРЕС
         PCHL ,          ;И ПЕРЕЙТИ ПО ЭТОМУ АДРЕСУ
```

2. Перейти по индексу, предполагая, что базовый адрес таблицы адресов находится в регистрах H и L, а индекс в аккумуляторе:

```
         ADD  A          ;УДВОИТЬ ИНДЕКС ДЛЯ 2-БАЙТНЫХ ЗАПИСЕЙ
         MOV  E,A        ;РАСШИРИТЬ ИНДЕКС ДО 16 РАЗРЯДОВ
         MVI  D,0
         DAD  D          ;ВЫЧИСЛИТЬ АДРЕС ЭЛЕМЕНТА
         MOV  E,M        ;ВЫБРАТЬ ЭЛЕМЕНТ ИЗ ТАБЛИЦЫ АДРЕСОВ
         INX  H
         MOV  D,M
         XCHG            ;ПЕРЕЙТИ ПО АДРЕСУ, ПОЛУЧЕННОМУ
         PCHL            ; ИЗ ТАБЛИЦЫ
```

Предполагается, что таблица адресов (таблица переходов) содержит не более 128 2-байтных записей, хранящихся в памяти в обычном для 8080, 8085 формате, при котором первым записан младший байт. Типовая таблица может иметь следующий вид:

```
JTAB:    DW   ROUT0      ;НАЧАЛЬНЫЙ АДРЕС ПРОГРАММЫ 0
         DW   ROUT1      ;НАЧАЛЬНЫЙ АДРЕС ПРОГРАММЫ 1
         DW   ROUT2      ;НАЧАЛЬНЫЙ АДРЕС ПРОГРАММЫ 2
```

3. Перейти и связать, т. е. передать управление по адресу DEST, сохранив текущее состояние счетчика команд в регистрах H и L:

```
         LXI  H,HERE     ;ЗАГРУЗИТЬ АДРЕС ДЛЯ СВЯЗИ В HL
HERE:    JMP  DEST       ;ПЕРЕДАТЬ УПРАВЛЕНИЕ
```

Эта процедура может обеспечить переход к подпрограмме без использования стека. Чтобы вернуть управление, подпрограмма может установить связь и выполнить команду PCHL. Например, чтобы вернуть управление команде, следующей сразу за JMP DEST, к H и L в подпрограмме следует добавить 3 (так как команда JMP DEST занимает три байта).

2.5.2. КОМАНДЫ УСЛОВНОГО ПЕРЕХОДА

1. Перейти при равенстве нулю.

• Перейти, если аккумулятор содержит 0:

```
         ANA  A          ;ПРОВЕРИТЬ АККУМУЛЯТОР
         JZ   DEST
```

• Перейти, если регистр содержит 0:

```
         INR  REG        ;ПРОВЕРИТЬ РЕГИСТР
         DCR  REG
         JZ   DEST
```

• Перейти, если ячейка памяти содержит 0:

```
         LXI  H,ADDR     ;ПРОВЕРИТЬ ЯЧЕЙКУ ПАМЯТИ
         INR  M
         DCR  M
         JZ   DEST
```

или

```
LDA  ADDR       ;ПРОВЕРИТЬ ЯЧЕЙКУ ПАМЯТИ
ANA  A
JZ   DEST
```

- Перейти, если пара регистров содержит 0:

```
MOV  A,RPH      ;ПРОВЕРИТЬ ПАРУ РЕГИСТРОВ
ORA  RPL
JZ   DEST
```

- Перейти, если разряд аккумулятора равен 0:

```
ANI  MASK       ;ПРОВЕРИТЬ РАЗРЯД АККУМУЛЯТОРА
JZ   DEST
```

Константа MASK содержит единицу в проверяемом разряде и нули во всех остальных. Заметим, что здесь происходит инверсия: если данный разряд аккумулятора равен 0, то в результате будет 0, а флаг нуля устанавливается в 1. Далее приводятся некоторые специальные случаи.

- Перейти, если разряд 7 аккумулятора равен 0:

```
RAL             ;ПЕРЕСЛАТЬ РАЗРЯД 7 ВО ФЛАГ ПЕРЕНОСА
JNC  DEST       ;И ПРОВЕРИТЬ ФЛАГ ПЕРЕНОСА
```

Можно заменить RAL на RLC, ADD A или ADC A:

- Перейти, если разряд 6 аккумулятора равен 0:

```
ADD  A          ;ПЕРЕСЛАТЬ РАЗРЯД 6 В ЗНАКОВЫЙ РАЗРЯД
JP   DEST       ;И ЗАТЕМ ПРОВЕРИТЬ ФЛАГ ЗНАКА
```

- Перейти, если разряд 0 аккумулятора равен 0:

```
RAR             ;ПЕРЕСЛАТЬ РАЗРЯД 0 ВО ФЛАГ ПЕРЕНОСА
JNC  DEST       ;И ПРОВЕРИТЬ ФЛАГ ПЕРЕНОСА
```

- Перейти, если данные последовательного ввода (SID) равны 0 (только в 8085):

```
RIM             ;ВЫБРАТЬ ПОСЛЕДОВАТЕЛЬНЫЙ ВВОД (SID)
RAL             ;ПЕРЕСЛАТЬ SID ВО ФЛАГ ПЕРЕНОСА
JNC  DEST
```

или

```
RIM             ;ВЫБРАТЬ ПОСЛЕДОВАТЕЛЬНЫЙ ВВОД (SID)
ANA  A          ;УСТАНОВИТЬ ФЛАГ ЗНАКА ПО SID
JP   DEST
```

- Перейти, если флаг разрешения прерываний (разряд 3 регистра I) равен 0 (только в 8085) [15]:

```
RIM             ;ПРОЧИТАТЬ РЕГИСТР МАСОК ПРЕРЫВАНИЙ
ANI  00001000B  ;ПРОВЕРИТЬ ФЛАГ РАЗРЕШЕНИЯ ПРЕРЫВАНИЙ
JZ   DEST
```

Данную последовательность команд можно использовать, чтобы сохранить текущее состояние системы прерываний перед выполнением команды при запрещенных прерываниях. Впоследствии это состояние может быть восстановлено.

• Перейти, если нет необработанных прерываний (разряды I7.5, I6.5 и I5.5 равны 0, только в 8085):

```
RIM             ;ПРОЧИТАТЬ РЕГИСТР МАСОК ПРЕРЫВАНИЙ
ANI   01110000B ;ВЫДЕЛИТЬ РАЗРЯДЫ НЕОБРАБОТАННЫХ
                ; ПРЕРЫВАНИЙ
JZ    DEST      ;ПЕРЕЙТИ, ЕСЛИ ВСЕ ОНИ РАВНЫ НУЛЮ
```

Разряды необработанных прерываний (I7.5, I6.5 и I5.5) – это соответственно разряды 6, 5 и 4 регистра масок прерываний (I).

2. Перейти при неравенстве нулю.

• Перейти, если аккумулятор не содержит 0:

```
ANA   A         ;ПРОВЕРИТЬ АККУМУЛЯТОР
JNZ   DEST
```

• Перейти, если регистр не содержит 0:

```
INR   REG       ;ПРОВЕРИТЬ РЕГИСТР
DCR   REG
JNZ   DEST
```

• Перейти, если ячейка памяти не содержит 0:

```
LXI   H,ADDR    ;ПРОВЕРИТЬ ЯЧЕЙКУ ПАМЯТИ
INR   M
DCR   M
JNZ   DEST
```

или

```
LDA   ADDR      ;ПРОВЕРИТЬ ЯЧЕЙКУ ПАМЯТИ
ANA   A
JNZ   DEST
```

• Перейти, если пара регистров не содержит 0:

```
MOV   A,RPH     ;ПРОВЕРИТЬ ПАРУ РЕГИСТРОВ
ORA   RPL
JNZ   DEST
```

• Перейти, если разряд аккумулятора равен 1:

```
ANI   MASK      ;ПРОВЕРИТЬ РАЗРЯД АККУМУЛЯТОРА
JNZ   DEST
```

Константа MASK содержит единицу в проверяемом разряде и нули во всех остальных. Заметим, что здесь происходит инверсия: если данный разряд аккумулятора равен 1, то в результате будет не 0 и флаг нуля очищается. Далее приводятся некоторые специальные случаи.

• Перейти, если разряд 7 аккумулятора равен 1:

```
RAL             ;ПЕРЕСЛАТЬ РАЗРЯД 7 ВО ФЛАГ ПЕРЕНОСА
JC    DEST      ;И ПРОВЕРИТЬ ФЛАГ ПЕРЕНОСА
```

Команду RAL можно заменить на RLC, ADD A или ADC A.

• Перейти, если разряд 6 аккумулятора равен 1:

```
ADD   A         ;ПЕРЕСЛАТЬ РАЗРЯД 6 В ЗНАКОВЫЙ РАЗРЯД
JM    DEST      ;И ЗАТЕМ ПРОВЕРИТЬ ФЛАГ ЗНАКА
```

• Перейти, если разряд 0 аккумулятора равен 1:

```
RAR                ;ПЕРЕСЛАТЬ РАЗРЯД 0 ВО ФЛАГ ПЕРЕНОСА
JC    DEST         ;И ПРОВЕРИТЬ ФЛАГ ПЕРЕНОСА
```

• Перейти, если данные последовательного ввода (SID) равны 1 (только в 8085):

```
RIM                ;ВЫБРАТЬ ПОСЛЕДОВАТЕЛЬНЫЙ ВВОД (SID)
RAL                ;ПЕРЕСЛАТЬ SID ВО ФЛАГ ПЕРЕНОСА
JC    DEST
```

или

```
RIM                ;ВЫБРАТЬ ПОСЛЕДОВАТЕЛЬНЫЙ ВВОД (SID)
ANA   A            ;УСТАНОВИТЬ ФЛАГ ЗНАКА ПО SID
JM    DEST
```

• Перейти, если флаг разрешения прерываний (разряд 3 регистра I) равен 1 (только в 8085) [15]:

```
RIM                ;ПРОЧИТАТЬ РЕГИСТР МАСОК ПРЕРЫВАНИЙ
ANI   00001000B    ;ПРОВЕРИТЬ ФЛАГ РАЗРЕШЕНИЯ ПРЕРЫВАНИЙ
JNZ   DEST
```

Данную последовательность команд можно использовать для того, чтобы сохранить текущее состояние системы прерываний перед выполнением команды при запрещенных прерываниях. Впоследствие это состояние может быть восстановлено.

• Перейти, если нет необработанных прерываний (не все разряды I7.5, I6.5 и I5.5 равны 0, только в 8085):

```
RIM                ;ПРОЧИТАТЬ РЕГИСТР МАСОК ПРЕРЫВАНИЙ
ANI   01110000B    ;ВЫДЕЛИТЬ РАЗРЯДЫ ОТЛОЖЕННЫХ ПРЕРЫВАНИЙ
JNZ   DEST         ;ПЕРЕЙТИ, ЕСЛИ ЕСТЬ ОТЛОЖЕННЫЕ ПРЕРЫВАНИЯ
```

Разряды необработанных прерываний (I7.5, I6.5 и I5.5) — это соответственно разряды 6, 5 и 4 регистра масок прерываний (I).

3. Перейти, если значения равны.

• Перейти, если (A) = VALUE:

```
CPI   VALUE        ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
JZ    DEST
```

Следующие два специальных варианта последовательностей команд применяются для любого регистра или при использовании H и L (регистр M) для ячейки памяти:

• Перейти, если (REG) = 1:

```
DCR   REG          ;ПРОВЕРИТЬ С ПОМОЩЬЮ УМЕНЬШЕНИЯ НА 1
JZ    DEST         ;И СРАВНЕНИЯ РЕЗУЛЬТАТА С НУЛЕМ
```

• Перейти, если (REG) = FF_{16}:

```
INR   REG          ;ПРОВЕРИТЬ С ПОМОЩЬЮ УВЕЛИЧЕНИЯ НА 1
JZ    DEST         ;И СРАВНЕНИЯ РЕЗУЛЬТАТА С НУЛЕМ
```

• Перейти, если (A) = (REG):

```
CMP   REG          ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
JZ    DEST
```

• Перейти, если (A) = (ADDR):

```
        LXI  H,ADDR     ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
        CMP  M
        JZ   DEST
```

• Перейти, если (RP) = VAL16:

```
        MOV  A,RPH      ;ПРОВЕРИТЬ СТАРШИЙ БАЙТ
        CPI  VAL16H
        JNZ  DONE
        MOV  A,RPL      ;ПРОВЕРИТЬ МЛАДШИЕ БАЙТЫ,
        CPI  VAL16L     ; ЕСЛИ ТОЛЬКО СТАРШИЕ БАЙТЫ РАВНЫ
        JZ   DEST       ;ПЕРЕЙТИ, ЕСЛИ ОБА БАЙТА РАВНЫ
DONE:   NOP
```

или

```
        LXI  H,-VAL16   ;HL = RP - VAL16
        DAD  RP
        MOV  A,H        ;ПРОВЕРИТЬ HL НА НУЛЬ
        ORA  L
        JZ   DEST
```

Использование команды DAD создает здесь некоторые трудности, так как она не влияет на флаг нуля, в результате чего HL необходимо проверять на наличие 0. Если RP есть HL, то первой командой должна быть LXI RP1, –VAL16, а второй DAD RP1.

• Перейти, если (RP1) = (RP2):

```
        MOV  A,RP1H     ;СРАВНИТЬ СТАРШИЕ БАЙТЫ
        CMP  RP2H
        JNZ  DONE
        MOV  A,RP1L     ;СРАВНИТЬ МЛАДШИЕ БАЙТЫ,
        CMP  RP2L       ; ЕСЛИ ТОЛЬКО СТАРШИЕ БАЙТЫ РАВНЫ
        JZ   DEST       ;ПЕРЕЙТИ, ЕСЛИ ОБА БАЙТА РАВНЫ
DONE:   NOP
```

Примечание: две следующие последовательности команд нельзя применять для проверки выхода за границы стека, так как операции, нарушившие эти границы, могли изменить указатель стека больше чем на 1:

• Перейти, если (SP) = VAL16:

```
        LXI  H,-VAL16   ;СФОРМИРОВАТЬ УКАЗАТЕЛЬ СТЕКА - VAL16
        DAD  SP
        MOV  A,H        ;ПРОВЕРИТЬ РАЗНОСТЬ НА НУЛЬ
        ORA  L
        JZ   DEST
```

• Перейти, если (SP) = (RP):

```
        LXI  H,0        ;ПЕРЕСЛАТЬ УКАЗАТЕЛЬ СТЕКА В HL
        DAD  SP
        MOV  A,H        ;СРАВНИТЬ СТАРШИЕ БАЙТЫ
        CMP  RPH
        JNZ  DONE
        MOV  A,L        ;ЕСЛИ СТАРШИЕ БАЙТЫ РАВНЫ,
        CMP  RPL        ; СРАВНИТЬ МЛАДШИЕ БАЙТЫ
        JZ   DEST
DONE:   NOP
```

4. Перейти, если значения не равны.

- Перейти, если (A) ≠ VALUE:

```
CPI  VALUE     ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
JNZ  DEST
```

Следующие два специальных варианта команд применяются для любого регистра или при использовании H и L (регистр M) для ячейки памяти:

- Перейти, если (REG) ≠ 1:

```
DCR  REG       ;ПРОВЕРИТЬ С ПОМОЩЬЮ УМЕНЬШЕНИЯ НА 1
JNZ  DEST      ; И СРАВНЕНИЯ РЕЗУЛЬТАТА С НУЛЕМ
```

- Перейти, если (REG) ≠ FF_{16}:

```
INR  REG       ;ПРОВЕРИТЬ С ПОМОЩЬЮ УВЕЛИЧЕНИЯ НА 1
JNZ  DEST      ; И СРАВНЕНИЯ РЕЗУЛЬТАТА С НУЛЕМ
```

- Перейти, если (A) ≠ (REG):

```
CMP  REG       ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
JNZ  DEST
```

- Перейти, если (A) ≠ (ADDR):

```
LXI  H,ADDR    ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
CMP  M
JNZ  DEST
```

- Перейти, если (RP) ≠ VAL16:

```
MOV  A,RPH     ;СРАВНИТЬ СТАРШИЕ БАЙТЫ
CPI  VAL16H
JNZ  DEST      ;ПЕРЕЙТИ, ЕСЛИ СТАРШИЕ БАЙТЫ НЕ РАВНЫ
MOV  A,RPL     ;ЕСЛИ СТАРШИЕ БАЙТЫ РАВНЫ,
CPI  VAL16L    ; СРАВНИТЬ МЛАДШИЕ БАЙТЫ
JNZ  DEST      ;ПЕРЕЙТИ, ЕСЛИ МЛАДШИЕ БАЙТЫ НЕ РАВНЫ
```

или

```
LXI  H,-VAL16  ;HL = RP - VAL16
DAD  RP
MOV  A,H       ;ПРОВЕРИТЬ РАЗНОСТЬ НА НУЛЬ
ORA  L
JNZ  DES
```

Использование команды DAD создает здесь некоторые трудности, так как она не влияет на флаг нуля, в результате чего необходимо HL проверять на равенство 0. Если RP есть HL, то первой командой должна быть LXI RP1, –VAL16, а второй DAD RP1.

Перейти, если (RP1) ≠ (RP2):

```
MOV  A,RP1H    ;СРАВНИТЬ СТАРШИЕ БАЙТЫ
CMP  RP2H
JNZ  DEST      ;ПЕРЕЙТИ, ЕСЛИ СТАРШИЕ БАЙТЫ НЕ РАВНЫ
MOV  A,RP1L    ;ЕСЛИ СТАРШИЕ БАЙТЫ РАВНЫ,
CMP  RP2L      ; СРАВНИТЬ МЛАДШИЕ БАЙТЫ
JNZ  DEST      ;ПЕРЕЙТИ, ЕСЛИ МЛАДШИЕ БАЙТЫ НЕ РАВНЫ
```

П р и м е ч а н и е: две следующие последовательности команд нельзя применять для проверки выхода за границы стека, так как операции, нарушившие эти границы, могли изменить указатель стека больше чем на 1.

• Перейти, если (SP) ≠ VAL16:

```
LXI  H,-VAL16   ;СФОРМИРОВАТЬ УКАЗАТЕЛЬ СТЕКА - VAL16
DAD  SP
MOV  A,H        ;ПРОВЕРИТЬ РАЗНОСТЬ НА НУЛЬ
ORA  L
JNZ  DEST
```

• Перейти, если (SP) ≠ (RP), где RP – пары регистров B и C или D и E:

```
LXI  H,0        ;ПЕРЕСЛАТЬ SP В HL
DAD  SP
MOV  A,H        ;СРАВНИТЬ СТАРШИЕ БАЙТЫ
CMP  RPH
JNZ  DEST       ;ПЕРЕЙТИ, ЕСЛИ СТАРШИЕ БАЙТЫ НЕ РАВНЫ
MOV  A,L        ;ЕСЛИ СТАРШИЕ БАЙТЫ РАВНЫ,
CMP  RPL        ; СРАВНИТЬ МЛАДШИЕ БАЙТЫ
JNZ  CEST
```

5. Перейти, если значение положительное.

• Перейти, если аккумулятор содержит положительное число:

```
ANA  A          ;ПРОВЕРИТЬ АККУМУЛЯТОР
JP   DEST
```

• Перейти, если регистр содержит положительное число:

```
INR  REG        ;ПРОВЕРИТЬ РЕГИСТР
DCR  REG
JP   DEST
```

• Перейти, если ячейка памяти содержит положительное число:

```
LXI  H,ADDR     ;ПРОВЕРИТЬ ЯЧЕЙКУ ПАМЯТИ
INR  M
DCR  M
JP   DEST
```

или

```
LDA  ADDR       ;ПРОВЕРИТЬ ЯЧЕЙКУ ПАМЯТИ
ANA  A          ; С ПОМОЩЬЮ A
JP   DEST
```

• Перейти, если пара регистров содержит положительное число:

```
INR  RPH        ;ПРОВЕРИТЬ ЗНАКОВЫЙ РАЗРЯД
DCR  RPH        ; СТАРШЕГО БАЙТА
JP   DEST
```

• Перейти, если 16-разрядное число в памяти ADDR и ADDR + 1 (старший байт в ADDR + 1) положительное:

```
LDA  ADDR+1     ;ПРОВЕРИТЬ ЗНАКОВЫЙ РАЗРЯД
ANA  A          ; СТАРШЕГО БАЙТА
JP   DEST
```

6. Перейти, если значение отрицательное.

• Перейти, если аккумулятор содержит отрицательное число:

```
ANA  A          ;ПРОВЕРИТЬ АККУМУЛЯТОР
JM   DEST
```

• Перейти, если регистр содержит отрицательное число:

```
INR  REG        ;ПРОВЕРИТЬ РЕГИСТР
DCR  REG
JM   DEST
```

• Перейти, если ячейка памяти содержит отрицательное число:

```
LXI  H,ADDR     ;ПРОВЕРИТЬ ЯЧЕЙКУ ПАМЯТИ
INR  M
DCR  M
JM   DEST
```

или

```
LDA  ADDR       ;ПРОВЕРИТЬ ЯЧЕЙКУ ПАМЯТИ
ANA  A          ; С ПОМОЩЬЮ A
JM   DEST
```

• Перейти, если пара регистров содержит отрицательное число:

```
INR  RPH        ;ПРОВЕРИТЬ ЗНАКОВЫЙ РАЗРЯД
DCR  RPH        ; СТАРШЕГО БАЙТА
JM   DEST
```

• Перейти, если ячейки памяти ADDR и ADDR + 1 (старший байт в ADDR + + 1) содержат отрицательное 16-разрядное число:

```
LDA  ADDR+1     ;ПРОВЕРИТЬ ЗНАКОВЫЙ РАЗРЯД
ANA  A          ; СТАРШЕГО БАЙТА
JM   DEST
```

7. Переходы с учетом знака.

Здесь предполагаем, что при переполнении для дополнения числа до двух выполняется та же работа, что была описана в гл. 1. Начальное сравнение всегда производится с помощью CPI VALUE (для 8-разрядных элементов данных) или CMP REG (где REG означает или регистр, или байт данных по адресу, содержащемуся в регистрах H и L).

• Перейти, если аккумулятор больше, чем VALUE (со знаком):

```
        CPI  VALUE      ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
        JM   DONE       ;ЕСЛИ РЕЗУЛЬТАТ ОТРИЦАТЕЛЬНЫЙ
        JNZ  DEST       ; ИЛИ РАВЕН НУЛЮ, ТО "НЕ БОЛЬШЕ, ЧЕМ"
DONE:   NOP
```

или

```
        CPI  VALUE+1    ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ ЗНАЧЕНИЯ + 1
        JP   DEST
```

Заметим, что при сравнении равных чисел флаг знака очищается.

• Перейти, если значение аккумулятора больше, чем регистра или ячейки памяти (со знаком):

```
        CMP   REG        ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
        JM    DONE       ;ЕСЛИ РЕЗУЛЬТАТ ОТРИЦАТЕЛЬНЫЙ
        JNZ   DEST       ; ИЛИ РАВЕН НУЛЮ, ТО "НЕ БОЛЬШЕ, ЧЕМ"
DONE:   NOP
```

или

```
        MOV   REG1,A     ;СФОРМИРОВАТЬ REG - A
        MOV   A,REG
        CMP   REG1
        JM    DEST       ;ПЕРЕЙТИ, ЕСЛИ РАЗНОСТЬ ОТРИЦАТЕЛЬНАЯ
```

или

```
        INR   REG        ;СФОРМИРОВАТЬ A - REG - 1
        CMP   REG
        JP    DEST       ;ПЕРЕЙТИ, ЕСЛИ РАЗНОСТЬ ПОЛОЖИТЕЛЬНАЯ
```

Проблема здесь состоит в том, чтобы избежать переход в случае равенства операндов. Третий из приведенных здесь вариантов команд изменяет регистр; если же заменить INR REG на DCR A, то изменяться будет только аккумулятор.

• Перейти, если аккумулятор больше или равен VALUE (со знаком):

```
        CPI   VALUE      ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
        JP    DEST       ;ПЕРЕЙТИ, ЕСЛИ РАЗНОСТЬ ПОЛОЖИТЕЛЬНАЯ
```

• Перейти, если аккумулятор больше или равен регистру или ячейке памяти (со знаком):

```
        CMP   REG        ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
        JP    DEST       ;ПЕРЕЙТИ, ЕСЛИ РАЗНОСТЬ ПОЛОЖИТЕЛЬНАЯ
```

• Перейти, если аккумулятор меньше, чем VALUE (со знаком):

```
        CPI   VALUE      ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
        JM    DEST       ;ПЕРЕЙТИ, ЕСЛИ РАЗНОСТЬ ОТРИЦАТЕЛЬНАЯ
```

• Перейти, если аккумулятор меньше, чем регистр или ячейка памяти (со знаком):

```
        CMP   REG        ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
        JM    DEST       ;ПЕРЕЙТИ, ЕСЛИ РАЗНОСТЬ ОТРИЦАТЕЛЬНАЯ
```

• Перейти, если аккумулятор меньше или равен VALUE (со знаком):

```
        CPI   VALUE      ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
        JM    DEST       ;ПЕРЕЙТИ, ЕСЛИ РАЗНОСТЬ ОТРИЦАТЕЛЬНАЯ
        JZ    DEST       ; ИЛИ НУЛЬ
```

или

```
        CPI   VALUE+1    ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ VALUE+1
        JM    DEST       ;ПЕРЕЙТИ, ЕСЛИ РАЗНОСТЬ ОТРИЦАТЕЛЬНАЯ
```

• Перейти, если аккумулятор меньше или равен ячейке памяти (со знаком):

```
        CMP   REG        ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
        JM    DEST       ;ПЕРЕЙТИ, ЕСЛИ РАЗНОСТЬ ОТРИЦАТЕЛЬНАЯ
        JZ    DEST       ; ИЛИ НУЛЬ
```

или

```
        MOV   REG1,A      ;СФОРМИРОВАТЬ REG- A
        MOV   A,REG
        CMP   REG1
        JP    DEST        ;ПЕРЕЙТИ, ЕСЛИ РАЗНОСТЬ ПОЛОЖИТЕЛЬНАЯ
```

или

```
        INR   REG         ;СФОРМИРОВАТЬ A - REG - 1
        CMP   REG
        JM    DEST        ;ПЕРЕЙТИ, ЕСЛИ РАЗНОСТЬ ОТРИЦАТЕЛЬНАЯ
```

В последнем случае мы могли бы заменить INR на DCR A, изменяя таким образом аккумулятор вместо регистра.

8. Перейти, если больше (без учета знака), т. е. если операнды не равны и при сравнении не требуется заема.

Особая проблема здесь состоит в том, чтобы избежать перехода в случае равенства операндов.

• Перейти, если (A) > VALUE (без знака):

```
        CPI   VALUE       ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
        JC    DONE        ;НЕ ПЕРЕХОДИТЬ, ЕСЛИ ТРЕБУЕТСЯ ЗАЕМ
        JNZ   DEST        ;ПЕРЕЙТИ, ЕСЛИ НЕТ ЗАЁМА И НЕТ РАВЕНСТВА
DONE:   NOP
```

При сравнении одинаковых чисел флаг переноса очищается.

Другой способ

```
        CPI   VALUE+1     ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ VALUE+1
        JNC   DEST        ;ПЕРЕЙТИ, ЕСЛИ ЗАЕМА НЕ ТРЕБУЕТСЯ
```

• Перейти, если (A) > (REG) (без знака):

```
        CMP   REG         ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
        JC    DONE        ;НЕ ПЕРЕХОДИТЬ, ЕСЛИ ТРЕБУЕТСЯ ЗАЕМ
        JNZ   DEST        ;ПЕРЕЙТИ, ЕСЛИ НЕТ ЗАЕМА И НЕТ РАВЕНСТВА
DONE:   NOP
```

или

```
        MOV   REG1,A      ;СФОРМИРОВАТЬ REG - A
        MOV   A,REG
        CMP   REG1
        JC    DEST        ;ПЕРЕЙТИ, ЕСЛИ ТРЕБУЕТСЯ ЗАЕМ
```

или

```
        INR   REG         ;СФОРМИРОВАТЬ A - REG - 1
        CMP   REG
        JNC   DEST        ;ПЕРЕЙТИ, ЕСЛИ ТРЕБУЕТСЯ ЗАЕМ
```

В третьем случае мы могли бы заменить INR REG на DCR A, изменяя таким образом аккумулятор вместо регистра.

• Перейти, если (A) > (ADDR) (без знака):

```
        LXI   H,ADDR
        CMP   M           ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
        JC    DONE        ;НЕ ПЕРЕХОДИТЬ, ЕСЛИ ЗАЕМА НЕ ТРЕБУЕТСЯ
        JNZ   DEST        ;ПЕРЕЙТИ, ЕСЛИ НЕТ ЗАЕМА И НЕТ РАВЕНСТВА
DONE:   NOP
```

или

```
MOV  REG,A        ;СФОРМИРОВАТЬ (ADDR) - A
LDA  ADDR
CMP  REG
JC   DEST         ;ПЕРЕЙТИ, ЕСЛИ ТРЕБУЕТСЯ ЗАЕМ
```

• Перейти, если (HL) > VAL16 (без знака):

```
LXI  RP,-(VAL16+1)   ;СФОРМИРОВАТЬ HL - VAL16 - 1
DAD  RP
JC   DEST            ;ПЕРЕЙТИ, ЕСЛИ ЗАЕМА НЕ ТРЕБУЕТСЯ
```

• Перейти, если (HL) > (RP) (без знака):

```
MOV  A,RPL        ;УСТАНОВИТЬ ЗАЕМ ПО МЛАДШИМ БАЙТАМ
CMP  L
MOV  A,RPH        ;ЗАТЕМ СРАВНИТЬ СТАРШИЕ БАЙТЫ,
SBB  H            ; УЧИТЫВАЯ ЗАЕМ
JC   DEST
```

• Перейти, если (SP) > VAL16 (без знака):

```
LXI  H,-(VAL16+1)    ;СФОРМИРОВАТЬ SP - VAL16 - 1
DAD  SP
JC   DEST            ;ПЕРЕЙТИ, ЕСЛИ ЗАЕМА НЕ ТРЕБУЕТСЯ
```

• Перейти, если (SP) > (RP) (без знака):

```
LXI  H,0          ;ПЕРЕСЛАТЬ SP В HL
DAD  SP
MOV  A,RPL        ;УСТАНОВИТЬ ЗАЕМ ПО МЛАДШИМ БАЙТАМ
CMP  L
MOV  A,RPH        ;ЗАТЕМ СРАВНИТЬ СТАРШИЕ БАЙТЫ,
SBB  H            ; УЧИТЫВАЯ ЗАЕМ
JC   DEST
```

9. Перейти, если значение не больше (без учета знака), т. е. если сравниваемые операнды равны или при их сравнении требуется заем.

Особая проблема здесь состоит в том, чтобы выполнить переход при равенстве операндов.

• Перейти, если (A) ⩽ VALUE (без знака):

```
CPI  VALUE        ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
JC   DEST         ;ПЕРЕЙТИ, ЕСЛИ ТРЕБУЕТСЯ ЗАЕМ
JZ   DEST         ; ИЛИ ЕСЛИ ОПЕРАНДЫ РАВНЫ
```

или

```
CPI  VALUE+1      ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ VALUE+1
JC   DEST         ;ПЕРЕЙТИ, ЕСЛИ ТРЕБУЕТСЯ ЗАЕМ
```

• Перейти, если (A) ⩽ REG (без знака):

```
CMP  REG          ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
JC   DEST         ;ПЕРЕЙТИ, ЕСЛИ ТРЕБУЕТСЯ ЗАЕМ
JZ   DEST         ; ИЛИ ЕСЛИ ОПЕРАНДЫ РАВНЫ
```

или

```
MOV  REG1,A       ;СФОРМИРОВАТЬ (REG) - (A)
MOV  A,REG
CMP  REG1
JNC  DEST         ;ПЕРЕЙТИ, ЕСЛИ ЗАЕМА НЕ ТРЕБУЕТСЯ
```

или

```
        INR   REG        ;СФОРМИРОВАТЬ A - REG - 1
        CMP   REG
        JC    DEST       ;ПЕРЕЙТИ, ЕСЛИ ЗАЕМА НЕ ТРЕБУЕТСЯ
```

В третьем случае мы могли бы заменить INR REG на DCR A, изменяя, таким образом, аккумулятор вместо регистра.

• Перейти, если (A) ≤ (ADDR) (без знака):

```
        LXI   H,ADDR
        CMP   M          ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
        JC    DEST       ;ПЕРЕЙТИ, ЕСЛИ ТРЕБУЕТСЯ ЗАЕМ
        JZ    DEST       ; ИЛИ ЕСЛИ ОПЕРАНДЫ РАВНЫ
```

или

```
        MOV   REG,A      ;СФОРМИРОВАТЬ (ADDR) - (A)
        LDA   ADDR
        CMP   REG
        JNC   DEST       ;ПЕРЕЙТИ, ЕСЛИ ЗАЕМА НЕ ТРЕБУЕТСЯ
```

• Перейти, если (HL) ≤ VAL16 (без знака):

```
        LXI   RP,-(VAL16+1)  ;СФОРМИРОВАТЬ HL - VAL16 - 1
        DAD   RP
        JNC   DEST           ;ПЕРЕЙТИ, ЕСЛИ ТРЕБУЕТСЯ ЗАЕМ
```

• Перейти, если (HL) ≤ (RP) (без знака):

```
        MOV   A,RPL      ;УСТАНОВИТЬ ЗАЕМ ПО МЛАДШИМ БАЙТАМ
        CMP   L
        MOV   A,RPH      ;ЗАТЕМ СРАВНИТЬ СТАРШИЕ БАЙТЫ,
        SBB   H          ; УЧИТЫВАЯ ЗАЕМ
        JNC   DEST
```

• Перейти, если (SP) ≤ VAL16 (без знака):

```
        LXI   H,-(VAL16+1)   ;СФОРМИРОВАТЬ SP - VAL16 - 1
        DAD   SP
        JNC   DEST           ;ПЕРЕЙТИ, ЕСЛИ ТРЕБУЕТСЯ ЗАЕМ
```

• Перейти, если (SP) ≤ (RP) (без знака):

```
        LXI   H,0        ;ПЕРЕСЛАТЬ SP В HL
        DAD   SP
        MOV   A,RPL      ;УСТАНОВИТЬ ЗАЕМ ПО МЛАДШИМ БАЙТАМ
        CMP   L
        MOV   A,RPH      ;ЗАТЕМ СРАВНИТЬ СТАРШИЕ БАЙТЫ
        SBB   H          ; УЧИТЫВАЯ ЗАЕМ
        JNC   DEST
```

10. Перейти, если значение меньше (без учета знака), т. е. если сравнение без знака требует заема.

• Перейти, если (A) < VALUE (без знака):

```
        CPI   VALUE      ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
        JC    DEST       ;ПЕРЕЙТИ, ЕСЛИ ТРЕБУЕТСЯ ЗАЕМ
```

• Перейти, если (A) < (REG) (без знака):

```
CMP  REG        ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
JC   DEST       ;ПЕРЕЙТИ, ЕСЛИ ТРЕБУЕТСЯ ЗАЕМ
```

• Перейти, если (A) < (ADDR) (без знака):

```
LXI  H,ADDR
CMP  M          ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
JC   DEST       ;ПЕРЕЙТИ, ЕСЛИ ТРЕБУЕТСЯ ЗАЕМ
```

• Перейти, если (HL) < VAL16 (без знака):

```
LXI  RP,-VAL16  ;СФОРМИРОВАТЬ HL - VAL16
DAD  RP
JNC  DEST       ;ПЕРЕЙТИ, ЕСЛИ ТРЕБУЕТСЯ ЗАЕМ
```

• Перейти, если (HL) < (RP) (без знака):

```
MOV  A,L        ;УСТАНОВИТЬ ЗАЕМ ПО МЛАДШИМ БАЙТАМ
CMP  RPL
MOV  A,H        ;ЗАТЕМ СРАВНИТЬ СТАРШИЕ БАЙТЫ,
SBB  RPH        ; УЧИТЫВАЯ ЗАЕМ
JC   DEST
```

• Перейти, если (SP) < VAL16 (без знака):

```
LXI  H,-VAL16   ;СФОРМИРОВАТЬ SP - VAL16
DAD  SP
JNC  DEST       ;ПЕРЕЙТИ, ЕСЛИ ТРЕБУЕТСЯ ЗАЕМ
```

• Перейти, если (SP) < (RP) (без знака):

```
LXI  H,0        ;ПЕРЕСЛАТЬ SP В HL
DAD  SP
MOV  A,L        ;УСТАНОВИТЬ ЗАЕМ ПО МЛАДШИМ БАЙТАМ
CMP  RPL
MOV  A,H        ;ЗАТЕМ СРАВНИТЬ СТАРШИЕ БАЙТЫ,
SBB  RPH        ; УЧИТЫВАЯ ЗАЕМ
JC   DEST
```

11. Перейти, если значение не меньше (без учета знака), т. е. если сравнение без знака не требует заема.

• Перейти, если (A) ⩾ VALUE (без знака):

```
CPI  VALUE      ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
JNC  DEST       ;ПЕРЕЙТИ, ЕСЛИ ЗАЕМА НЕ ТРЕБУЕТСЯ
```

• Перейти, если (A) ⩾ (REG) (без знака):

```
CMP  REG        ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
JNC  DEST       ;ПЕРЕЙТИ, ЕСЛИ ЗАЕМА НЕ ТРЕБУЕТСЯ
```

• Перейти, если (A) ⩾ (ADDR) (без знака):

```
LXI  H,ADDR     ;СРАВНИТЬ С ПОМОЩЬЮ ВЫЧИТАНИЯ
CMP  M
JNC  DEST       ;ПЕРЕЙТИ, ЕСЛИ ЗАЕМА НЕ ТРЕБУЕТСЯ
```

• Перейти, если (HL) ≥ VAL16 (без знака):

```
LXI  RP,-VAL16 ;СФОРМИРОВАТЬ HL - VAL16
DAD  RP
JC   DEST      ;ПЕРЕЙТИ, ЕСЛИ ЗАЕМА НЕ ТРЕБУЕТСЯ
```

• Перейти, если (HL) ≥ (RP) (без знака):

```
MOV  A,L       ;УСТАНОВИТЬ ЗАЕМ ПО МЛАДШИМ БАЙТАМ
CMP  RPL
MOV  A,H       ;ЗАТЕМ СРАВНИТЬ СТАРШИЕ БАЙТЫ,
SBB  RPH       ; УЧИТЫВАЯ ЗАЕМ
JNC  DEST
```

• Перейти, если (SP) ≥ VAL16 (без знака):

```
LXI  H,-VAL16  ;СФОРМИРОВАТЬ SP - VAL16
DAD  SP
JC   DEST      ;ПЕРЕЙТИ, ЕСЛИ ЗАЕМА НЕ ТРЕБУЕТСЯ
```

• Перейти, если (SP) ≥ (RP) (без знака):

```
LXI  H,0       ;ПЕРЕСЛАТЬ SP В HL
DAD  SP
MOV  A,L       ;УСТАНОВИТЬ ЗАЕМ ПО МЛАДШИМ БАЙТАМ
CMP  RPL
MOV  A,H       ;ЗАТЕМ СРАВНИТЬ СТАРШИЕ БАЙТЫ,
SBB  RPH       ; УЧИТЫВАЯ ЗАЕМ
JNC  DEST
```

2.6. КОМАНДЫ ПРОПУСКА

В микропроцессорах 8080 или 8085 команда пропуска может быть выполнена с помощью команды перехода с соответствующим адресом назначения. Этот адрес назначения должен указывать на команду, следующую после той, которая стоит непосредственно за командой перехода. Действительное число пропускаемых байтов будет меняться, так как команды микропроцессоров 8080 и 8085 могут иметь длину 1 – 3 байта.

2.7. КОМАНДЫ ВЫЗОВА ПОДПРОГРАММ

2.7.1. КОМАНДЫ БЕЗУСЛОВНОГО ВЫЗОВА

В микропроцессорах 8080 или 8085 косвенный вызов может быть выполнен с помощью обращения к промежуточной подпрограмме, которая переходит косвенно на вызываемую подпрограмму. Команда RET в конце вызываемой подпрограммы передаст затем управление в начальную точку вызова. Основная программа выполняет команду

```
CALL TRANS
```

а подпрограмма TRANS передает управление в конечный адрес назначения. Заметим, что TRANS заканчивается командой перехода, а не возврата. Типичная подпрограмма TRANS выглядит следующим образом.

• Для вызова по адресу, содержащемуся в паре регистров H и L:

```
TRANS:    PCHL              ;ТОЧКА ВХОДА СОДЕРЖИТСЯ В HL
```

• Для вызова по адресу, содержащемуся в паре регистров D и E:

```
TRANS:    XCHG              ;ТОЧКА ВХОДА СОДЕРЖИТСЯ В DE
          PCHL
```

• Для вызова по адресу, содержащемуся в паре регистров B и C:

```
TRANS:    MOV  H,B          ;ТОЧКА ВХОДА СОДЕРЖИТСЯ В BC
          MOV  L,C
          PCHL
```

или

```
TRANS:    PUSH B            ;ТОЧКА ВХОДА СОДЕРЖИТСЯ В BC
          RET
```

Второй способ реализуется медленнее первого, но при нем H и L остаются без изменения.

• Для вызова по адресу, содержащемуся в ячейках памяти ADDR и ADDR + + 1:

```
TRANS:    LHLD ADDR         ;ТОЧКА ВХОДА СОДЕРЖИТСЯ В ADDR
          PCHL
```

• Для вызова по адресу, содержащемуся в вершине стека. Здесь необходимо обменять адрес возврата и вершину стека. Это может быть сделано в основной программе следующим образом:

```
          LXI  H,RETPT      ;ВЗЯТЬ АДРЕС ТОЧКИ ВОЗВРАТА
          XTHL              ;ЗАПИСАТЬ АДРЕС ВОЗВРАТА В СТЕК
          PCHL              ;И ПЕРЕЙТИ ПО СТАРОМУ АДРЕСУ,
                            ; НАХОДИВШЕМУСЯ В ВЕРШИНЕ СТЕКА
```

Такой обмен может позволить возобновить затем выполнение приостановленной программы или же обеспечить специальный выход на подпрограмму обработки ошибок.

Вызовы подпрограмм по индексу могут быть выполнены точно так же, как и косвенные вызовы. Команда CALL передает управление промежуточной подпрограмме, выполняющей индексный переход. Эта подпрограмма заканчивается командой безусловного перехода (обычно PCHL), которая не оказывает влияния на стек. Команда RET в конце вызываемой подпрограммы передает управление назад в первоначальную точку вызова.

Если основная программа выполняет команду CALL JMPIND и при этом индекс находится в аккумуляторе, а начальный адрес таблицы переходов — в регистрах H и L, то подпрограмма индексного перехода выглядит следующим образом:

```
JMPIND:   ADD  A            ;УДВОИТЬ ИНДЕКС ДЛЯ 2-БАЙТНЫХ ЗАПИСЕЙ
          MOV  E,A          ;РАСШИРИТЬ ИНДЕКС ДО 16 РАЗРЯДОВ
          MVI  D,0
          DAD  D            ;ВЫЧИСЛИТЬ АДРЕС ЭЛЕМЕНТА
          MOV  E,M          ;ВЫБРАТЬ ЭЛЕМЕНТ ИЗ ТАБЛИЦЫ ПЕРЕХОДОВ
          INX  H
          MOV  D,M
          XCHG              ;И ПЕРЕДАТЬ УПРАВЛЕНИЕ ПО АДРЕСУ
          PCHL              ; ЭТОГО ЭЛЕМЕНТА
```

Одна из проблем, встречающихся при организации индексных и косвенных вызовов, состоит в том, что вспомогательная подпрограмма может нарушить выполнение вызываемой подпрограммы. Например, подпрограмма индексного перехода JMPIND изменяет аккумулятор, регистры D и E, регистры H и L и флаги. Таким образом, ни один из этих регистров не может быть использован для передачи параметров вызываемой подпрограмме. Программист всегда должен помнить, что промежуточная подпрограмма перехода вклинивается между основной программой и вызываемой подпрограммой. Подобная же вставка происходит в тех случаях, когда программы операционной системы передают управление от одной задачи к другой или из основной программы к драйверу ввода-вывода или программе обработки прерываний.

2.7.2. КОМАНДЫ УСЛОВНОГО ВЫЗОВА

В микропроцессоре 8080 или 8085 условный вызов подпрограммы может быть выполнен с помощью показанных ранее последовательностей команд для условных переходов. Единственное отличие состоит в том, что команды перехода к действительным адресам назначения должны быть заменены на команды вызова подпрограммы (например, JNZ DEST заменяется на CNZ DEST или JP DEST на CP DEST).

2.8. КОМАНДЫ ВОЗВРАТА ИЗ ПОДПРОГРАММ

2.8.1. КОМАНДЫ БЕЗУСЛОВНОГО ВОЗВРАТА

Команда RET автоматически возвращает управление по адресу, который хранится в вершине стека. Если адрес возврата находится где-либо в другом месте (например, в паре регистров в двух фиксированных ячейках памяти), то управление может быть передано по этому адресу с помощью выполнения косвенного перехода.

2.8.2. КОМАНДЫ УСЛОВНОГО ВОЗВРАТА

В микропроцессоре 8080 или 8085 условные возвраты из подпрограммы могут быть выполнены с помощью последовательностей команд, показанных ранее для условных переходов. Единственное отличие состоит в том, что переходы к действительным адресам назначения должны быть заменены на команды возврата из подпрограммы (например, JNC DEST заменяется на RNC или JM DEST на RM).

2.8.3. КОМАНДЫ ВОЗВРАТА С ПРОПУСКОМ

• Добавить фиксированное значение смещения к адресу, хранящемуся в вершине стека, перед возвратом управления по этому адресу. Эта последовательность команд позволяет программисту передать управление командам, следующим после параметров, данных и других невыполняемых элементов программы:

```
 POP  D          ;ВЗЯТЬ АДРЕС ВОЗВРАТА
 LXI  H,OFFSET   ;ДОБАВИТЬ К АДРЕСУ ВОЗВРАТА СМЕЩЕНИЕ
 DAD  D
 PCHL            ;ВОЗВРАТИТЬСЯ С ПРОПУСКОМ
```

• Изменить адрес возврата на RETPT. Предполагается, что текущий адрес возврата хранится в вершине стека:

```
LXI  H,RETPT    ;ИЗМЕНИТЬ АДРЕС ВОЗВРАТА НА RETPT
XTHL
```

Команда XTHL обменивает пару регистров HL с вершиной стека. Эта процедура может использоваться для организации специального выхода из подпрограммы обработки ошибок или специальной обработки без изменения логики подпрограммы или потери исходного адреса возврата.

2.8.4. КОМАНДЫ ВОЗВРАТА ПОСЛЕ ПРЕРЫВАНИЯ

Если в начале программы обработки прерываний все регистры сохраняются с помощью последовательности команд

```
PUSH PSW        ;СОХРАНИТЬ ВСЕ РЕГИСТРЫ
PUSH B
PUSH D
PUSH H
```

то стандартная последовательность для возврата (в конце этой программы) выглядит следующим образом:

```
POP  H          ;ВОССТАНОВИТЬ ВСЕ РЕГИСТРЫ
POP  D
POP  B
POP  PSW
EI              ;ВНОВЬ РАЗРЕШИТЬ ПРЕРЫВАНИЯ
RET
```

Регистры должны восстанавливаться в порядке, обратном тому, в котором они сохранялись. Во избежание излишней записи адресов возврата в стек команда EI должна стоять непосредственно перед RET.

2.9. СМЕШАННЫЕ КОМАНДЫ

В эту категорию входят следующие команды: нет операции, запись в стек, получение из стека, останов, ожидание, захват (программное прерывание) и другие, не попавшие в описанные ранее категории команд.

1. Команда "нет операции".

Как и NOP, любая команда MOV с одним и тем же источником и назначением не выполняет ничего, кроме увеличения программного счетчика. К этим дополнительным командам "нет операции" относятся следующие:

```
MOV  A,A
MOV  B,B
MOV  C,C
MOV  D,D
MOV  E,E
MOV  H,H
MOV  L,L
```

Команда MOV M, M отсутствует.

2. Команды записи в стек.

• Записать в стек один регистр (A, B, D или H)

```
PUSH RP         ;ЗАПИСАТЬ В СТЕК ДВОЙНОЙ РЕГИСТР
INX  SP         ;НО ЗАТЕМ УДАЛИТЬ МЛАДШИЙ БАЙТ
```

В качестве RP может выступать PSW, т. е. слово состояния процессора (содержащее аккумулятор и флаги). Программисты обычно предпочитают комбинировать операнды длиной в байт или просто терять байт стека вместо того, чтобы пытаться записать в стек один регистр.

• Записать в стек регистр маски прерываний (регистр I, только в 8085):

```
RIM             ;ПЕРЕСЛАТЬ I В A
PUSH PSW        ;ЗАПИСАТЬ I И F В СТЕК
INX  SP         ;УДАЛИТЬ F ИЗ СТЕКА
```

Эта последовательность команд позволяет сохранить флаг разрешения прерываний (разряд 3 регистра I) для его последующего восстановления.

• Записать в стек ячейку памяти ADDR:

```
LDA  ADDR       ;ПОЛУЧИТЬ ДАННЫЕ ИЗ ПАМЯТИ
PUSH PSW        ;СОХРАНИТЬ ДАННЫЕ, ФЛАГИ
INX  SP         ;ЗАТЕМ УДАЛИТЬ ФЛАГИ
```

Ячейка памяти ADDR может быть внешним регистром приоритета или управления (или копией внешнего регистра).

• Записать в стек ячейки памяти ADDR и ADDR + 1:

```
LHLD ADDR       ;ЗАПИСАТЬ В СТЕК ЯЧЕЙКИ ПАМЯТИ
PUSH H
```

3. Команды чтения из стека.

• Прочитать из стека один регистр (A, B, D или H), предполагая, что он был сохранен в стеке так, как это было показано ранее:

```
DCX  SP         ;ВЕРНУТЬ НАЗАД УКАЗАТЕЛЬ СТЕКА
POP  RP         ;ПРОЧИТАТЬ ИЗ СТЕКА ДВОЙНОЙ РЕГИСТР
```

Эта последовательность команд изменяет младшую по значению половину пары регистров непредсказуемым образом.

• Прочитать из стека регистр масок прерываний (регистр I), предполагая, что он был сохранен в вершине стека (только в 8085):

```
DCX  SP         ;ВЕРНУТЬ НАЗАД УКАЗАТЕЛЬ СТЕКА
POP  PSW        ;ПРОЧИТАТЬ ИЗ СТЕКА A И ФЛАГИ
```

Нельзя сразу после этих команд восстановить регистр I с помощью команды SIM, так как значение этого регистра при записи отличается от значения, получаемого при чтении командой RIM. Значение, прочитанное из стека, может быть использовано следующим образом.

• Восстановить маску прерываний (разряды 0, 1 и 2):

```
ANI  00000111B ;ВЫДЕЛИТЬ РАЗРЯДЫ МАСОК ПРЕРЫВАНИЙ
ORI  00001000B ;РАЗРЕШИТЬ УСТАНОВКУ МАСОК ПРЕРЫВАНИЙ
SIM            ;ВОССТАНОВИТЬ РАЗРЯДЫ МАСОК ПРЕРЫВАНИЙ
```

Заметим, что для изменения маски прерываний должен быть установлен разряд 3 (разрешить установку маски прерываний).

• Разрешить прерывания, если ранее они были разрешены (разряд 3 = 1):

```
       ANI  00001000B ;ДО ЭТОГО ПРЕРЫВАНИЯ БЫЛИ РАЗРЕШЕНЫ?
       JZ   DONE
       EI             ;ДА, РАЗРЕШИТЬ ИХ ВНОВЬ
DONE:  NOP
```

• Прочитать из стека ячейку памяти ADDR, предполагая, что она было сохранена в вершине стека:

```
DCX  SP        ;ВЕРНУТЬ НАЗАД УКАЗАТЕЛЬ СТЕКА
POP  PSW       ;ПОЛУЧИТЬ ИЗ СТЕКА АККУМУЛЯТОР И ФЛАГИ
STA  ADDR      ;ВОССТАНОВИТЬ ДАННЫЕ В ПАМЯТИ
```

Эта последовательность команд изменяет флаги непредсказуемым образом. Ячейка памяти ADDR может быть внешним регистром управления или приоритета (или копией внешнего регистра).

• Прочитать из стека ячейки памяти ADDR и ADDR + 1, предполагая, что они были сохранены в стеке так, как было показано ранее:

```
POP  H         ;ЗАПИСАТЬ В СТЕК ЯЧЕЙКИ ПАМЯТИ
SHLD ADDR
```

Иногда помимо регистров необходимо записывать в стек и читать из стека какие-либо ключевые ячейки памяти и другие величины.

2.9.1. КОМАНДЫ ОЖИДАНИЯ

В микропроцессоре 8080 или 8085 выполнить ожидание проще всего с помощью бесконечного цикла:

```
HERE:     JMP  HERE      ;ЖДАТЬ НА МЕСТЕ
```

Процессор будет выполнять команду JMP до тех пор, пока не произойдет прерывания, и вновь будет ее выполнять после того, как программа обработки прерываний возвратит управление. В микропроцессоре 8080 прерывания должны быть разрешены с помощью команды EI, иначе процессор будет выполнять цикл бесконечно. В микропроцессоре 8085 с помощью команды TRAP (которая не маскируется) прерывание может быть распознано без разрешения прерываний.

2.9.2. КОМАНДА ЗАХВАТА

В микропроцессорах 8080 и 8085 захваты (называемые также программными прерываниями) выполняются с помощью команд RST (см. табл. 1.10). Команда RST N вызывает подпрограмму, начинающуюся с адреса 8 × N. Таким образом, RST 0, например, после сохранения в стеке счетчика команд передает управление по адресу памяти 0000. Аналогично RST 6 после сохранения счетчика команд в стеке передает управление по адресу 0030_{16}. Обычно команды RST используются системой прерываний, однако программист может назначить некоторые неиспользуемые прерывания для общих подпрограмм, захватов ошибок или в качестве точек входа в супервизор. Тогда команда RST служит однобайтной командой вызова подпрограммы.

2.9.3. КОМАНДЫ КОРРЕКЦИИ

1. Перейти, если аккумулятор не содержит правильное десятичное число (в коде BCD):

```
MOV  REG,A     ;СОХРАНИТЬ А
ADI  0         ;ПОСЛЕ ПРИБАВЛЕНИЯ 0
DAA            ; КОРРЕКТИРОВАТЬ А В ДЕСЯТИЧНЫЙ ВИД
```

```
        CMP   REG        ;ДЕСЯТИЧНАЯ КОРРЕКЦИЯ ИЗМЕНИЛА A?
        JNZ   DEST       ;ДА, ЗНАЧИТ, ЭТО НЕ БЫЛО ПРАВИЛЬНОЕ
                         ; ДЕСЯТИЧНОЕ ЧИСЛО
```

2. Десятичное увеличение аккумулятора на 1 (к аккумулятору добавить 1 в десятичном виде):

```
        ADI   1
        DAA
```

3. Десятичное уменьшение аккумулятора на 1 (из аккумулятора вычесть 1 в десятичном виде):

```
        ADI   99H
        DAA
```

2.9.4. КОМАНДЫ РАЗРЕШЕНИЯ ПРЕРЫВАНИЙ

1. Разрешить прерывания, если разряд 3 сохраненного регистра I равен 1 (только в 8085):

```
        POP   PSW        ;ВОССТАНОВИТЬ СОХРАНЕННЫЙ РЕГИСТР I
        ANI   00001000B  ;ПРЕРЫВАНИЯ БЫЛИ РАЗРЕШЕНЫ?
        JZ    DONE
        EI               ;ДА, РАЗРЕШИТЬ ИХ ВНОВЬ
DONE:   NOP
```

Эта последовательность команд позволяет вновь разрешить прерывания после того, как процессор выполнил некоторую последовательность команд при запрещенных прерываниях. Смысл этой процедуры заключается в том, чтобы избежать ненужного разрешения прерываний в конце этой последовательности, если изначально они были запрещены.

При программировании для микропроцессора 8080 проблема состоит в том, что у этого микропроцессора отсутствует флаг разрешения прерываний, который можно было бы прочитать [15]. Программист может решить эту проблему, создав в ячейке ОЗУ флаг разрешения прерываний. Например, если эта ячейка называется IFLAG, то вместо обычных команд DI и EI должна использоваться соответственно одна из последовательностей команд:

```
        DI               ;ЗАПРЕТИТЬ ПРЕРЫВАНИЯ
        LXI   H,IFLAG    ;И ОЧИСТИТЬ В ОЗУ ФЛАГ
        MVI   M,0        ; РАЗРЕШЕНИЯ ПРЕРЫВАНИЙ
```

и

```
        LXI   H,IFLAG    ;УСТАНОВИТЬ В ОЗУ ФЛАГ
        MVI   M,1        ; РАЗРЕШЕНИЯ ПРЕРЫВАНИЙ
        EI               ;РАЗРЕШИТЬ ПРЕРЫВАНИЯ
```

Теперь IFLAG можно использовать для определения того, разрешены или запрещены в данный момент прерывания. Программа, которая должна выполняться при запрещенных прерываниях (но вход в нее может произойти как при разрешенных, так и при запрещенных прерываниях) должна в начале сохранить IFLAG и запретить прерывания, а в конце восстановить IFLAG и, если IFLAG указывает, что перед началом выполнения прерывания были разрешены, вновь разрешить прерывания. Заметим, что ячейка IFLAG устанавливается и очищается при запрещенных прерываниях.

2. Размаскировать (разрешить) прерывания RST 5.5, RST 6.5 и RST 7.5 – только в 8085.

• Размаскировать RST 5.5:

```
MVI  A,00001110B    ;РАЗМАСКИРОВАТЬ RST 5.5
SIM
```

или

```
RIM                ;ПРОЧИТАТЬ СТАРЫЕ МАСКИ ПРЕРЫВАНИЙ
ANI  00000110B     ;СОХРАНИТЬ RST 6.5, RST 7.5
ORI  00001000B     ;РАЗРЕШИТЬ УСТАНОВКУ МАСОК
SIM                ;РАЗМАСКИРОВАТЬ RST 5.5
```

Вторая последовательность команд сохраняет остальные маски прерываний.

• Размаскировать RST 6.5:

```
MVI  A,00001101B    ;РАЗМАСКИРОВАТЬ RST 6.5
SIM
```

или

```
RIM                ;ПРОЧИТАТЬ СТАРЫЕ МАСКИ ПРЕРЫВАНИЙ
ANI  00000101B     ;СОХРАНИТЬ RST 5.5, RST 7.5
ORI  00001000B     ;РАЗРЕШИТЬ УСТАНОВКУ МАСОК
SIM                ;РАЗМАСКИРОВАТЬ RST 6.5
```

Вторая последовательность команд сохраняет остальные маски прерываний.

• Размаскировать RST 7.5:

```
MVI  A,00001011B    ;РАЗМАСКИРОВАТЬ RST 7.5
SIM
```

или

```
RIM                ;ПРОЧИТАТЬ СТАРЫЕ МАСКИ ПРЕРЫВАНИЙ
ANI  00000011B     ;СОХРАНИТЬ RST 5.5, RST 6.5
ORI  00001000B     ;РАЗРЕШИТЬ УСТАНОВКУ МАСОК
SIM                ;РАЗМАСКИРОВАТЬ RST 7.5
```

Вторая последовательность команд сохраняет остальные маски прерываний.

2.9.5. КОМАНДЫ ЗАПРЕЩЕНИЯ ПРЕРЫВАНИЙ

1. Маскировать (запретить) прерывание RST 5.5, RST 6.5 и RST 7.5 – только в 8085.

• Маскировать RST 5.5:

```
MVI  A,00001111B    ;МАСКИРОВАТЬ RST 5.5
SIM
```

или

```
RIM                ;ПРОЧИТАТЬ СТАРЫЕ МАСКИ ПРЕРЫВАНИЙ
ANI  00000111B     ;ВЫДЕЛИТЬ РАЗРЯДЫ МАСОК
ORI  00001001B     ;РАЗРЕШИТЬ УСТАНОВКУ МАСОК
                   ; И УСТАНОВИТЬ МАСКУ RST 5.5
SIM                ;МАСКИРОВАТЬ RST 5.5
```

Вторая последовательность команд сохраняет остальные маски прерываний.

• Маскировать RST 6.5:

```
MVI  A,00001111B    ;МАСКИРОВАТЬ RST 6.5
SIM
```

или

```
        RIM                   ;ПРОЧИТАТЬ СТАРЫЕ МАСКИ ПРЕРЫВАНИЙ
        ANI   00000111B       ;ВЫДЕЛИТЬ РАЗРЯДЫ МАСОК
        ORI   00001010B       ;РАЗРЕШИТЬ УСТАНОВКУ МАСОК
                              ; И УСТАНОВИТЬ МАСКУ RST 6.5
        SIM                   ;МАСКИРОВАТЬ RST 6.5
```

Вторая последовательность команд сохраняет остальные маски прерываний.

• Маскировать RST 7.5:

```
        MVI   A,00001111B     ;МАСКИРОВАТЬ RST 7.5
        SIM
```

или

```
        RIM                   ;ПРОЧИТАТЬ СТАРЫЕ МАСКИ ПРЕРЫВАНИЙ
        ANI   00000111B       ;ВЫДЕЛИТЬ РАЗРЯДЫ МАСОК
        ORI   00001100B       ;РАЗРЕШИТЬ УСТАНОВКУ МАСОК
                              ; И УСТАНОВИТЬ МАСКУ RST 7.5
        SIM                   ;МАСКИРОВАТЬ RST 7.5
```

Вторая последовательность команд сохраняет остальные маски прерываний.

• Очистить (сбросить) RST 7.5:

```
        MVI   A,00010000B     ;СБРОСИТЬ ТРИГГЕР RST 7.5
        SIM
```

Положительный (возрастающий) фронт при RST 7.5 всегда устанвливает триггер RST 7.5, даже если это прерывание было замаскировано, или запрещено.

• Очистить (сбросить) сигнал RST 7.5, если он активен (ожидает обработки):

```
        RIM                   ;ПРОЧИТАТЬ МАСКИ ПРЕРЫВАНИЙ
        ANI   01000000B       ;RST 7.5 ОЖИДАЕТ ОБРАБОТКИ?
        JZ    DONE
        MVI   A,00010000B     ;ДА, СБРОСИТЬ ТРИГГЕР RST 7.5
        SIM
DONE:   NOP
```

Сигнал RST 7.5 сбрасывается с помощью команды SIM при установленном разряде 4 аккумулятора. Так как разряды 3 и 6 аккумулятора равны 0, то эта команда SIM, не воздействует на маски прерываний или последовательный ввод данных (SOD).

2.9.6. КОМАНДЫ ТРАНСЛЯЦИИ

1. Транслировать аккумулятор в соответствующую запись таблицы, адрес начала которой содержится в регистрах H и L:

```
        MOV   E,A             ;РАСШИРИТЬ ОПЕРАНД ДО 16-РАЗРЯДНОГО
        MVI   D,0             ; ИНДЕКСА
        DAD   D               ;ИСПОЛЬЗОВАТЬ ОПЕРАНД ДЛЯ ДОСТУПА
                              ; К ТАБЛИЦЕ
        MOV   A,M             ;ЗАМЕНИТЬ ОПЕРАНД НА ЗАПИСЬ ТАБЛИЦЫ
```

Эта процедура может переводить данные из одного кода в другой.

2. Транслировать аккумулятор в соответствующую 16-разрядную запись таблицы, адрес начала которой содержится в регистрах H и L. Поместить запись в регистры H и L:

```
XCHG           ;ПЕРЕСЛАТЬ БАЗОВЫЙ АДРЕС В DE
MOV  L,A       ;РАСШИРИТЬ ОПЕРАНД ДО 16-РАЗРЯДНОГО
MVI  H,0       ; ИНДЕКСА
DAD  H         ;УДВОИТЬ ИНДЕКС ДЛЯ 2-БАЙТНЫХ ЗАПИСЕЙ
DAD  D         ;ВЫЧИСЛИТЬ ИНДЕКСНЫЙ АДРЕС
MOV  E,M       ;ПОЛУЧИТЬ ЗАПИСЬ
INX  H
MOV  D,M
XCHG           ;ПЕРЕСЛАТЬ ЗАПИСЬ В HL
```

Применение DAD для удваивания операнда позволяет использовать в качестве операнда любую 8-разрядную величину (при использовании ADD A пришлось бы ограничиться значением, меньшим 128).

2.9.7. СМЕШАННЫЕ КОМАНДЫ

1. Зарезервировать область памяти в стеке; уменьшить указатель стека с тем, чтобы обеспечить NUM свободных ячеек в вершине стека:

```
LXI  H,-NUM    ;ДОБАВИТЬ NUM БАЙТ К ВЕРШИНЕ СТЕКА
DAD  SP
SPHL           ;SP = SP - NUM
```

Альтернативой является последовательность команд DCX SP.

2. Отказаться от области в стеке; увеличить указатель стека, чтобы удалить NUM временных ячеек из вершины стека:

```
LXI  H,NUM     ;УДАЛИТЬ NUM БАЙТ ИЗ ВЕРШИНЫ СТЕКА
DAD  SP
SPHL           ;SP = SP + NUM
```

Альтернативой является последовательность команд INX SP.

2.10. ДОПОЛНИТЕЛЬНЫЕ СПОСОБЫ АДРЕСАЦИИ

• **Косвенная адресация**. В процессорах 8080 и 8085 косвенную адресацию можно выполнить с помощью загрузки косвенных адресов в регистры H и L, используя команду LHLD. После этого обращение к регистру M является эквивалентом косвенной адресации. Таким образом, этот процесс всегда включает два шага. Кроме того, можно использовать также пары регистров B и D в командах LDAX и STAX.

Примеры

1. Загрузить аккумулятор косвенно из ячейки памяти, адрес которой содержится в ячейках памяти ADDR и ADDR + 1:

```
LHLD ADDR      ;ВЫБРАТЬ КОСВЕННЫЙ АДРЕС
MOV  A,M       ;ВЫБРАТЬ ДАННЫЕ КОСВЕННО
```

2. Запомнить аккумулятор косвенно по адресу, который находится в ADDR и ADDR + 1:

```
LHLD ADDR      ;ВЫБРАТЬ КОСВЕННЫЙ АДРЕС
MOV  M,A       ;ЗАПОМНИТЬ ДАННЫЕ КОСВЕННО
```

3. Загрузить аккумулятор косвенно из ячейки памяти, адрес которой содержится в регистрах H и L:

```
MOV  E,M       ;ВЫБРАТЬ КОСВЕННЫЙ АДРЕС
INX  H
```

```
MOV  D,M
LDAX D          ;ВЫБРАТЬ ДАННЫЕ КОСВЕННО
```

4. Запомнить аккумулятор косвенно в ячейке памяти, адрес которой содержится в регистрах H и L:

```
MOV  E,M        ;ВЫБРАТЬ КОСВЕННЫЙ АДРЕС
INX  H
MOV  D,M
STAX D          ;ЗАПОМНИТЬ ДАННЫЕ КОСВЕННО
```

5. Перейти косвенно по адресу, который содержится в ADDR и ADDR + 1:

```
LHLD ADDR       ;ВЫБРАТЬ КОСВЕННЫЙ АДРЕС
PCHL            ;И ПЕРЕЙТИ ПО ЭТОМУ АДРЕСУ
```

Косвенные обращения могут повторяться неограниченно, когда необходимо обеспечить многоуровневую косвенную адресацию. Например, в следующей программе для загрузки аккумулятора косвенно используется косвенный адрес:

```
MOV  E,M        ;ВЫБРАТЬ ПЕРВЫЙ КОСВЕННЫЙ АДРЕС
INX  H
MOV  D,M
XCHG
MOV  E,M        ;ИСПОЛЗОВАТЬ КОСВЕННЫЙ АДРЕС КОСВЕННО
INX  H
MOV  D,M
LDAX D          ;ВЫБРАТЬ ДАННЫЕ КОСВЕННО
```

Косвенные адреса должны запоминаться в памяти в обычном для 8080, 8085 формате, т. е. младший байт записывается первым (по меньшему адресу).

• **Индексная адресация.** Индексную адресацию можно выполнить, добавляя индекс с помощью команды DAD к базе. Понятно, что программное сложение требует дополнительного времени выполнения.

Примеры.

1. Загрузить аккумулятор из ячейки памяти, заданной индексным адресом, получаемым прибавлением аккумулятора к фиксированному базовому адресу:

```
LXI  D,BASE     ;ВЗЯТЬ БАЗОВЫЙ АДРЕС
MOV  L,A        ;РАСШИРИТЬ ИНДЕКС ДО 16 РАЗРЯДОВ
MVI  H,0
DAD  D          ;ВЫЧИСЛИТЬ ИНДЕКСНЫЙ АДРЕС
MOV  A,M        ;ВЫБРАТЬ ДАННЫЕ ПО ИНДЕКСНОМУ АДРЕСУ
```

2. Загрузить аккумулятор из ячейки памяти, заданной индексным адресом, получаемым прибавлением аккумулятора к ячейкам памяти BASE и BASE + 1:

```
LHLD BASE       ;ВЗЯТЬ БАЗОВЫЙ АДРЕС
MOV  E,A        ;РАСШИРИТЬ ИНДЕКС ДО 16 РАЗРЯДОВ
MVI  D,0
DAD  D          ;ВЫЧИСЛИТЬ ИНДЕКСНЫЙ АДРЕС
MOV  A,M        ;ВЫБРАТЬ ДАННЫЕ ПО ИНДЕКСНОМУ АДРЕСУ
```

3. Загрузить аккумулятор из ячейки памяти, заданной индексным адресом, получаемым прибавлением ячеек памяти INDEX и INDEX + 1 к регистрам H и L:

```
XCHG            ;СОХРАНИТЬ БАЗУ В DE
LHLD INDEX      ;ВЗЯТЬ ИНДЕКС ИЗ ПАМЯТИ
DAD  D          ;ВЫЧИСЛИТЬ ИНДЕКСНЫЙ АДРЕС
MOV  A,M        ;ВЫБРАТЬ ДАННЫЕ ПО ИНДЕКСНОМУ АДРЕСУ
```

4. Перейти по индексу к команде перехода в списке команд. Индекс находится в аккумуляторе, а базовый адрес списка – в регистрах H и L:

```
MOV  B,A        ;УТРОИТЬ ИНДЕКС ДЛЯ 3-БАЙТНОЙ
ADD  A          ; КОМАНДЫ JMP
ADD  B
MOV  C,A        ;РАСШИРИТЬ УТРОЕННЫЙ ИНДЕКС
MVI  B,0        ; ДО 16 РАЗРЯДОВ
DAD  B          ;ВЫЧИСЛИТЬ ИНДЕКСНЫЙ АДРЕС
PCHL            ;И ПЕРЕДАТЬ ТУДА УПРАВЛЕНИЕ
```

Список, начинающийся с адреса BASE обычно выглядит следующим образом:

```
BASE:   JMP  SUB0       ;ПЕРЕЙТИ К ПОДПРОГРАММЕ 0
        JMP  SUB1       ;ПЕРЕЙТИ К ПОДПРОГРАММЕ 1
        JMP  SUB2       ;ПЕРЕЙТИ К ПОДПРОГРАММЕ 2
        ...
```

Так как команда JMP занимает три байта, то перед добавлением индекса к базовому адресу следует индекс умножить на 3. Если длина списка больше 256 байт, то для умножения на 3 можно использовать следующую процедуру:

```
XCHG            ;СОХРАНИТЬ БАЗУ В DE
MOV  L,A        ;РАСШИРИТЬ ИНДЕКС ДО 16 РАЗРЯДОВ
MVI  H,0
MOV  B,L        ;СКОПИРОВАТЬ ИНДЕКС В BC
MOV  C,H
DAD  H          ;УДВОИТЬ ИНДЕКС
DAD  B          ;УТРОИТЬ ИНДЕКС
DAD  D          ;ВЫЧИСЛИТЬ ИНДЕКСНЫЙ АДРЕС
PCHL            ; И ПЕРЕЙТИ ПО НЕМУ
```

• **Предувеличение.** При предувеличении адресный регистр перед использованием автоматически увеличивается. В процессоре 8080 или 8085 предувеличение может быть реализовано с помощью увеличения пары регистров перед ее использованием в качестве адреса.

Примеры.

1. Загрузить аккумулятор, используя предувеличение регистров H и L:

```
INX  H          ;УВЕЛИЧИТЬ H И L ДО ИСПОЛЬЗОВАНИЯ
                ; В КАЧЕСТВЕ АДРЕСА
MOV  A,M        ;ВЫБРАТЬ ДАННЫЕ
```

2. Запомнить аккумулятор, используя предувеличение регистров D и E:

```
INX  D          ;УВЕЛИЧИТЬ D И E ДО ИСПОЛЬЗОВАНИЯ
                ; В КАЧЕСТВЕ АДРЕСА
STAX D          ;ЗАПОМНИТЬ ДАННЫЕ
```

3. Загрузить регистры D и E из памяти, начиная с адреса, превышающего на 2 содержимое регистров H и L:

```
INX  H          ;УВЕЛИЧИТЬ H И L НА 2 ДО ИСПОЛЬЗОВАНИЯ
INX  H          ; В КАЧЕСТВЕ АДРЕСА
```

```
MOV  E,M          ;ВЫБРАТЬ МЛАДШИЙ БАЙТ
INX  H
MOV  D,M          ;ВЫБРАТЬ СТАРШИЙ БАЙТ
```

Предувеличение на 2 необходимо при работе с массивами адресов или 16-разрядных элементов данных.

4. Запомнить аккумулятор в ячейках памяти ADDR и ADDR + 1, используя предувеличение:

```
LHLD ADDR         ;УВЕЛИЧИТЬ H И L ДО ИСПОЛЬЗОВАНИЯ
INX  H            ; В КАЧЕСТВЕ АДРЕСА
MOV  M,A          ;ЗАПОМНИТЬ ДАННЫЕ
SHLD ADDR         ;ОБНОВИТЬ КОСВЕННЫЙ АДРЕС
```

Можно комбинировать предувеличение с косвенной адресацией. Здесь ячейки памяти ADDR и ADDR + 1 могли бы указывать на последнюю занятую ячейку в буфере.

5. Передать управление команде, адрес которой на 2 больше, чем содержимое ячеек памяти NXTPGM и NXTPGM + 1:

```
LHLD NXTPGM       ;ВЗЯТЬ УКАЗАТЕЛЬ
INX  H            ;УВЕЛИЧИТЬ УКАЗАТЕЛЬ ДО ИСПОЛЬЗОВАНИЯ
INX  H            ; В КАЧЕСТВЕ АДРЕСА
SHLD NXTPGM       ;ОБНОВИТЬ УКАЗАТЕЛЬ
MOV  E,M          ;ВЫБРАТЬ НАЧАЛЬНЫЙ АДРЕС, ИСПОЛЬЗУЯ
INX  H            ; УКАЗАТЕЛЬ
MOV  D,M
XCHG              ;И ПЕРЕДАТЬ УПРАВЛЕНИЕ ПО ЭТОМУ АДРЕСУ
PCHL
```

Здесь NXTPGW и NXTPGM + 1 указывают на начальный адрес программы, которую процессор выполнял перед этим. Обычно NXTPGM и NXTPGM + 1 содержат BASE – 2, где BASE является начальным адресом таблицы программ. Типичная таблица обычно выглядит следующим образом:

```
BASE:     DW   ROUT0       ;НАЧАЛЬНЫЙ АДРЕС ПРОГРАММЫ 0
          DW   ROUT1       ;НАЧАЛЬНЫЙ АДРЕС ПРОГРАММЫ 1
          DW   ROUT2       ;НАЧАЛЬНЫЙ АДРЕС ПРОГРАММЫ 2
          DW   ROUT3       ;НАЧАЛЬНЫЙ АДРЕС ПРОГРАММЫ 3
          ...
```

• **Послеувеличение.** При послеувеличении адресный регистр после использования в команде автоматически увеличивается. В процессоре 8080 или 8085 послеувеличение может быть реализовано с помощью увеличения пары регистров после ее использования в качестве адреса. Заметим, что подобным образом процессор увеличивает указатель стека при выполнении команд POP и RET.

Примеры

1. Загрузить аккумулятор, используя послеувеличение регистров H и L:

```
MOV  A,M          ;ВЫБРАТЬ ДАННЫЕ
INX  H            ;УВЕЛИЧИТЬ H И L ПОСЛЕ ИСПОЛЬЗОВАНИЯ
                  ; В КАЧЕСТВЕ АДРЕСА
```

2. Запомнить аккумулятор, используя послеувеличение регистров D и E:

```
STAX D            ;ЗАПОМНИТЬ ДАННЫЕ
INX  D            ;УВЕЛИЧИТЬ D И E ПОСЛЕ ИСПОЛЬЗОВАНИЯ
                  ; В КАЧЕСТВЕ АДРЕСА
```

3. Загрузить регистры D и E из ячеек памяти, начальный адрес которых содержится в регистрах H и L. После этого увеличить содержимое регистров H и L на 2:

```
MOV  E,M        ;ВЫБРАТЬ МЛАДШИЙ БАЙТ
INX  H
MOV  D,M        ;ВЫБРАТЬ СТАРШИЙ БАЙТ
INX  H
```

Послеувеличение на 2 необходимо при работе с массивами адресов или 16-разрядных элементов. Заметим, что послеувеличение обычно проще и естественнее, чем предувеличение.

4. Запомнить аккумулятор, используя послеувеличение ячеек памяти ADDR и ADDR + 1:

```
LHLD ADDR       ;ВЫБРАТЬ КОСВЕННЫЙ АДРЕС
MOV  M,A        ;ЗАПОМНИТЬ ДАННЫЕ
INX  H          ;УВЕЛИЧИТЬ КОСВЕННЫЙ АДРЕС ПОСЛЕ
SHLD ADDR       ; ИСПОЛЬЗОВАНИЯ
```

Послеувеличение можно комбинировать с косвенной адресацией. Здесь ячейки памяти ADDR и ADDR + 1 могли бы указывать на следующую свободную ячейку в буфере.

5. Передать управление по адресу, хранящемуся в ячейках памяти с адресом NXTPGM и NXTPGM + 1. Затем увеличить эти ячейки памяти на 2:

```
LHLD NXTPGM
MOV  E,M        ;ВЫБРАТЬ НАЧАЛЬНЫЙ АДРЕС, ИСПОЛЬЗУЯ
INX  H          ; УКАЗАТЕЛЬ
MOV  D,M
INX  H          ;ЗАКОНЧИТЬ УВЕЛИЧЕНИЕ АДРЕСА
SHLD NXTPGM     ; ПОСЛЕ ИСПОЛЬЗОВАНИЯ
XCHG            ;ПЕРЕДАТЬ УПРАВЛЕНИЕ ПО НАЧАЛЬНОМУ АДРЕСУ
PCHL
```

Здесь NXTPGM и NXTPGM + 1 указывают на начальный адрес следующей программы, которую должен выполнять процессор. Обычно NXTPGM и NXTPGM + 1 вначале содержат BASE, т. е. начальный адрес таблицы программ. Типичная таблица обычно выглядит следующим образом:

```
BASE:     DW   ROUT0     ;НАЧАЛЬНЫЙ АДРЕС ПРОГРАММЫ 0
          DW   ROUT1     ;НАЧАЛЬНЫЙ АДРЕС ПРОГРАММЫ 1
          DW   ROUT2     ;НАЧАЛЬНЫЙ АДРЕС ПРОГРАММЫ 2
          DW   ROUT3     ;НАЧАЛЬНЫЙ АДРЕС ПРОГРАММЫ 3
          ...
```

• **Предуменьшение.** При предуменьшении адресный регистр перед использованием автоматически уменьшается. В процессоре 8080 или 8085 предуменьшение может быть выполнено с помощью уменьшения пары регистров перед ее использованием в качестве адреса. Заметим, что подобным образом процессор уменьшает указатель стека при выполнении команд PUSH и CALL.

Примеры

1. Загрузить аккумулятор, используя предуменьшение регистров H и L:

```
DCX  H          ;УМЕНЬШИТЬ H И L ПЕРЕД ИСПОЛЬЗОВАНИЕМ
                ; В КАЧЕСТВЕ АДРЕСА
MOV  A,M        ;ВЫБРАТЬ ДАННЫЕ
```

2. Запомнить аккумулятор, используя предуменьшение регистров D и E:

```
DCX  D          ;УМЕНЬШИТЬ D И E ПЕРЕД ИСПОЛЬЗОВАНИЕМ
                ; В КАЧЕСТВЕ АДРЕСА
STAX D          ;ЗАПОМНИТЬ ДАННЫЕ
```

3. Загрузить регистры D и E из памяти, начиная с адреса, на 2 меньшего, чем содержимое регистров H и L:

```
DCX  H          ;ВЫБРАТЬ СТАРШИЙ БАЙТ
MOV  D,M
DCX  H          ;ВЫБРАТЬ МЛАДШИЙ БАЙТ
MOV  E,M
```

Предуменьшение на 2 необходимо при работе с массивами адресов или 16-разрядных элементов данных. Заметим, что предуменьшение обычно проще и естественнее, чем послеуменьшение.

4. Запомнить аккумулятор, используя предуменьшение ячеек памяти ADDR и ADDR + 1:

```
LHLD ADDR       ;УМЕНЬШИТЬ КОСВЕННЫЙ АДРЕС
DCX  H          ; ПЕРЕД ИСПОЛЬЗОВАНИЕМ
MOV  M,A        ;ЗАПОМНИТЬ ДАННЫЕ
SHLD ADDR       ;ОБНОВИТЬ КОСВЕННЫЙ АДРЕС
```

Предуменьшение можно комбинировать с косвенной адресацией. Здесь ячейки памяти ADDR и ADDR + 1 могли бы указывать на последнюю занятую ячейку в стеке.

5. Передать управление по адресу, хранящемуся по адресу, на 2 меньшему, чем содержимое ячеек памяти NXTPGM и NXTPGM +1:

```
LHLD NXTPGM     ;ВЫБРАТЬ КОСВЕННЫЙ АДРЕС,
DCX  H          ; ИСПОЛЬЗУЯ УКАЗАТЕЛЬ
MOV  D,M
DCX  H
MOV  E,M
SHLD NXTPGM     ;ЗАПОМНИТЬ УМЕНЬШЕННЫЙ УКАЗАТЕЛЬ
XCHG            ;ПЕРЕДАТЬ УПРАВЛЕНИЕ НА НАЧАЛЬНЫЙ
PCHL            ; АДРЕС
```

Здесь NXTPGM и NXTPGM + 1 указывают на начальный адрес в списке самой последней выполненной программы. Вначале NXTPGM и NXTPGM + 1 обычно содержат FINAL + 2, где FINAL – адрес последней записи в таблице программ. Типичная таблица обычно выглядит следующим образом:

```
        DW   ROUT0      ;НАЧАЛЬНЫЙ АДРЕС ПРОГРАММЫ 0
        DW   ROUT1      ;НАЧАЛЬНЫЙ АДРЕС ПРОГРАММЫ 1
        .
        .
        .
FINAL:  DW   ROUTL      ;НАЧАЛЬНЫЙ АДРЕС ПОСЛЕДНЕЙ ПРОГРАММЫ
```

Работа с таблицей происходит в обратном порядке. Этот подход полезен при вычислении по математическим формулам, которые вводятся с клавиатуры. Например, если ЭВМ должна вычислить выражение

```
Z = LN (A X SIN (B X EXP (C X Y)))
```

то она должна работать от конца к началу, т. е. порядок операций будет следующим:

1) вычислить C × Y,
2) вычислить EXP (C × Y),
3) вычислить B × EXP (C × Y),
4) Вычислить SIN (B × EXP(C × Y)),
5) вычислить A × SIN (B × EXP (C × Y)),
6) вычислить LN (A × SIN (B × EXP (C × Y))).

Работа в обратном направлении удобна, когда ЭВМ не может начинать выполнение задачи, пока не получит полную строку или команду. В этом случае она должна работать от конца к началу.

• **Послеуменьшение.** При послеуменьшении адресный регистр после использования автоматически уменьшается. В процессоре 8080 или 8085 послеуменьшение может быть выполнено с помощью уменьшения пары регистров после использования ее в качестве адреса.

Примеры

1. Загрузить аккумулятор, используя послеуменьшение регистров H и L:

```
MOV  A,M        ;ВЫБРАТЬ ДАННЫЕ
DCX  H          ;УМЕНЬШИТЬ H И L ПОСЛЕ ИСПОЛЬЗОВАНИЯ
                ; В КАЧЕСТВЕ АДРЕСА
```

2. Запомнить аккумулятор, используя послеуменьшение регистров D и E:

```
STAX D          ;ЗАПОМНИТЬ ДАННЫЕ
DCX  D          ;УМЕНЬШИТЬ D И E ПОСЛЕ ИСПОЛЗЬОВАНИЯ
                ; В КАЧЕСТВЕ АДРЕСА
```

3. Загрузить регистры D и E из ячеек памяти, начальный адрес которых содержится в регистрах H и L. После этого уменьшить H и L на 2:

```
INX  H          ;ВЫБРАТЬ СТАРШИЙ БАЙТ
MOV  D,M
DCX  H          ;ВЫБРАТЬ МЛАДШИЙ БАЙТ
MOV  E,M
DCX  H          ;УМЕНЬШИТЬ HL НА 2 ПОСЛЕ ИСПОЛЬЗОВАНИЯ
DCX  H          ; В КАЧЕСТВЕ АДРЕСА
```

Послеуменьшение на 2 необходимо при работе с массивами адресов или 16-разрядных элементов данных.

4. Запомнить аккумулятор, используя послеуменьшение ячеек памяти ADDR и ADDR + 1:

```
LHLD ADDR       ;ВЫБРАТЬ КОСВЕННЫЙ АДРЕС
MOV  M,A        ;ЗАПОМНИТЬ ДАННЫЕ
DCX  H          ;УМЕНЬШИТЬ КОСВЕННЫЙ АДРЕС
SHLD ADDR       ; ПОСЛЕ ИСПОЛЬЗОВАНИЯ
```

Послеуменьшение можно комбинировать с косвенной адресацией. Здесь ячейки памяти ADDR и ADDR + 1 могли бы указывать на следующую свободную ячейку в стеке.

5. Передать управление по адресу, хранящемуся в ячейках памяти NXTPGM и NXTPGM + 1. Затем уменьшить эти ячейки на 2:

```
LHLD NXTPGM     ;ВЫБРАТЬ УКАЗАТЕЛЬ
INX  H          ;ВЫБРАТЬ НАЧАЛЬНЫЙ АДРЕС
```

```
MOV  D,M
DCX  H
MOV  E,M
DCX  H           ;УМЕНЬШИТЬ УКАЗАТЕЛЬ ПОСЛЕ
DCX  H           ; ИСПОЛЬЗОВАНИЯ
SHLD NXTPGM
XCHG             ;ПЕРЕЙТИ НА НАЧАЛЬНЫЙ АДРЕС
PCHL
```

Здесь NXTPGM и NXTPGM + 1 указывают на адрес следующей программы, которую должен выполнять процессор. Вначале NXTPGM и NXTPGM + 1 обычно содержат FINAL, т. е. адрес последней записи в таблице программ. Типичная таблица обычно выглядит следующим образом:

```
        DW   ROUT0     ;НАЧАЛЬНЫЙ АДРЕС ПРОГРАММЫ 0
        DW   ROUT1     ;НАЧАЛЬНЫЙ АДРЕС ПРОГРАММЫ 1
        .
        .
        .
FINAL:  DW   ROUTL     ;НАЧАЛЬНЫЙ АДРЕС ПОСЛЕДНЕЙ ПРОГРАММЫ
```

Здесь ЭВМ работает с таблицей в обратном порядке. Этот подход удобен при интерпретации команд, введенных с клавиатуры обычным способом слева-направо. Например, предположим, что оператор системы управления процессом вводит команду SET TEMP (POSITION 2) = MEAN (TEMP (POSITION 1), TEMP (POSITION 3)).

Программа системы управления должна выполнять команду, работая справа налево и начиная с содержимого внутренних скобок;

- Определить индекс, соответствующий POSITION 1.
- Получить TEMP (POSITION 1) из таблицы измерения температур.
- Определить индекс, соответствующий POSITION 3.
- Получить TEMP (POSITION 3) из таблицы измерения температур.
- Вычислить MEAN (TEMP (POSITION 1), TEMP (POSITION 3)) с помощью выполнения программы MEAN с двумя входными величинами в качестве параметров.
- Определить индекс, соответствующий POSITION 2.
- Выполнить функцию SET, которая, возможно, включает в себя установку регулятора и параметры для достижения желаемого значения TEMP (POSITION 2).

Оператор вводит команду слева направо, от внешних скобок к внутренним. С другой стороны, ЭВМ должна выполнять команду изнутри наружу (начиная с внутренних скобок) и справа налево. Очевидно, что автоматическое уменьшение адреса является удобным способом реализации такого изменения направления обработки.

- **Косвенная адресация с предварительным индексированием (предындексирование).** При предындексировании процессор должен сначала вычислить индексный адрес, а затем использовать этот адрес косвенно. Так как таблица, для которой производится индексирование, должна содержать двухбайтные косвенные адреса, индексирование должно сопровождаться умножением на 2.

Примеры

• Загрузить аккумулятор, используя предындексирование. Базовый адрес содержится в регистрах H и L, а индекс – в аккумуляторе:

```
ADD  A          ;УДВОИТЬ ИНДЕКС ДЛЯ 2-БАЙТНЫХ ЗАПИСЕЙ
MOV  E,A        ;РАСШИРИТЬ ИНДЕКС ДО 16 РАЗРЯДОВ
MVI  D,0
DAD  D          ;ВЫЧИСЛИТЬ ИНДЕКСНЫЙ АДРЕС
MOV  E,M        ;ПОЛУЧИТЬ КОСВЕННЫЙ АДРЕС
INX  H
MOV  D,M
LDAX D          ;ПОЛУЧИТЬ ДАННЫЕ КОСВЕННО
```

• Запомнить аккумулятор, используя предындексирование. Базовый адрес находится в ячейках памяти ADDR и ADDR + 1, а индекс – в ячейке памяти INDEX:

```
LHLD ADDR       ;ВЫБРАТЬ БАЗОВЫЙ АДРЕС
MOV  B,A        ;СОХРАНИТЬ ДАННЫЕ
LDA  INDEX      ;ВЫБРАТЬ ИНДЕКС
ADD  A          ;УДВОИТЬ ИНДЕКС ДЛЯ 2-БАЙТНЫХ ЗАПИСЕЙ
MOV  E,A        ;РАСШИРИТЬ ИНДЕКС ДО 16 РАЗРЯДОВ
MVI  D,0
DAD  D          ;ВЫЧИСЛИТЬ ИНДЕКСНЫЙ АДРЕС
MOV  E,M        ;ПОЛУЧИТЬ КОСВЕННЫЙ АДРЕС
INX  H
MOV  D,M
XCHG            ;ЗАПОМНИТЬ ДАННЫЕ КОСВЕННО
MOV  M,B
```

• Передать управление (перейти) по адресу, получаемому косвенно из таблицы, начинающейся с адреса JTAB. Индекс находится в аккумуляторе:

```
ADD  A          ;УДВОИТЬ ИНДЕКС ДЛЯ 2-БАЙТНЫХ ЗАПИСЕЙ
MOV  E,A        ;РАСШИРИТЬ ИНДЕКС ДО 16 РАЗРЯДОВ
MVI  D,0
LXI  A,JTAB     ;ВЗЯТЬ БАЗОВЫЙ АДРЕС
DAD  D          ;ВЫЧИСЛИТЬ ИНДЕКСНЫЙ АДРЕС
MOV  E,M        ;ПОЛУЧИТЬ КОСВЕННЫЙ АДРЕС
INX  H
MOV  D,M
XCHG            ;ПЕРЕДАТЬ УПРАВЛЕНИЕ ПО КОСВЕННОМУ
PCHL            ; АДРЕСУ
```

Таблица, начинающаяся по адресу JTAB, может выглядеть следующим образом:

```
JTAB:   DW   ROUT0      ;НАЧАЛЬНЫЙ АДРЕС ПРОГРАММЫ 0
        DW   ROUT1      ;НАЧАЛЬНЫЙ АДРЕС ПРОГРАММЫ 1
        DW   ROUT2      ;НАЧАЛЬНЫЙ АДРЕС ПРОГРАММЫ 2
        ...
```

• **Косвенная адресация с последующим индексированием (послеиндексирование)**. При послеиндексировании процессор должен сначала получить косвенный адрес, а затем использовать его как базу для индексирования.

Примеры

• Загрузить аккумулятор, используя послеиндексирование. Базовый ад-

рес находится в ячейках памяти ADDR и ADDR + 1, а индекс – в аккумуляторе:

```
LHLD ADDR      ;ПОЛУЧИТЬ БАЗОВЫЙ АДРЕС КОСВЕННО
MOV  E,A       ;РАСШИРИТЬ ИНДЕКС ДО 16 РАЗРЯДОВ
MVI  D,0
DAD  D         ;ВЫЧИСЛИТЬ ИНДЕКСНЫЙ АДРЕС
MOV  A,M
```

• Запомнить аккумулятор, используя послеиндексирование. Базовый адрес находится в ячейках памяти ADDR и ADDR + 1, а индекс – в ячейке памяти INDEX:

```
LHLD ADDR      ;ПОЛУЧИТЬ БАЗОВЫЙ АДРЕС КОСВЕННО
MOV  B,A       ;СОХРАНИТЬ ДАННЫЕ
LDA  INDEX     ;ПОЛУЧИТЬ ИНДЕКС
MOV  E,A       ;РАСШИРИТЬ ИНДЕКС ДО 16 РАЗРЯДОВ
MVI  D,0
DAD  D         ;ВЫЧИСЛИТЬ ИНДЕКСНЫЙ АДРЕС
MOV  M,B       ;ЗАПОМНИТЬ ДАННЫЕ
```

При изменении содержимого ячеек памяти ADDR и ADDR + 1 эта программа может быть пригодна для работы с различными массивами.

• Передать управление (перейти) по адресу, получаемому с помощью индексирования из базового адреса, содержащегося в ячейках памяти ADDR и ADDR + 1. Индекс содержится в аккумуляторе:

```
MOV  B,A       ;УТРОИТЬ ИНДЕКС ДЛЯ 3-БАЙТНЫХ
ADD  A         ; КОМАНД JMP
ADD  B
MOV  E,A       ;РАСШИРИТЬ ИНДЕКС ДО 16 РАЗРЯДОВ
MVI  D,0
LHLD ADDR      ;ПОЛУЧИТЬ БАЗОВЫЙ АДРЕС КОСВЕННО
DAD  D         ;ВЫЧИСЛИТЬ ИНДЕКСНЫЙ АДРЕС
PCHL           ;И ПЕРЕДАТЬ НА НЕГО УПРАВЛЕНИЕ
```

Таблица содержит 3-байтные команды JMP; типовой пример:

```
BASE:   JMP  ROUT0     ;ПЕРЕЙТИ К ПРОГРАММЕ 0
        JMP  ROUT1     ;ПЕРЕЙТИ К ПРОГРАММЕ 1
        JMP  ROUT2     ;ПЕРЕЙТИ К ПРОГРАММЕ 2
        ...
```

Адрес BASE должен быть помещен в ячейки памяти ADDR и ADDR + 1.

ГЛАВА 3

РАСПРОСТРАНЕННЫЕ ОШИБКИ ПРОГРАММИРОВАНИЯ

В этой главе описываются распространенные ошибки в программах на языке ассемблера 8080 и 8085. Заключительный раздел данной главы посвящен описанию часто встречающихся ошибок в драйверах ввода-вывода и программах обслуживания прерываний. Эта глава преследует следующие цели:

предупредить программистов о возможных неприятных местах и источниках ошибок,

описать вероятные источники ошибок программирования,

подчеркнуть те методы и предостережения, которые обсуждались в гл. 1 и 2,

информировать программистов, занимающихся поддержкой математического обеспечения, о возможных местах ошибок и неправильных толкований,

дать начинающему программисту отправную точку в трудном процессе обнаружения и исправления ошибок.

Конечно, никакой список ошибок не может быть полным. Ранее уже обращалось внимание на большинство распространенных ошибок, за исключением тех редких и трудноуловимых, которые ставят в тупик даже опытного программиста. Тем не менее многие совершенно очевидные ошибки остались нерассмотренными, и данное в этой главе описание поможет читателю отлаживать большинство программ.

3.1. КЛАССИФИКАЦИЯ ОШИБОК ПРОГРАММИРОВАНИЯ

Распространенные ошибки программирования для микропроцессоров 8080 и 8085 могут быть разделены на следующие категории:

• Перестановка операндов или частей операндов. К типичным ошибкам этого рода относятся перестановка операндов, указывающих на источник и назначение в командах пересылки, перевертывание формата, в котором запоминаются 16-разрядные значения, изменение направления при вычитаниях и сравнениях.

• Неправильное использование флагов. Типичные ошибки следующие: использование не того флага, который в данном конкретном случае должен проверяться (как, например, флага знака вместо флага переноса), условный переход после команд, которые не воздействуют на данный флаг, инвертирование условий перехода (особенно при использовании флага нуля), неправильный условный переход в случаях равенства и случайное изменение флага перед условным переходом.

• Смешивание регистров и пар регистров. Типичная ошибка состоит в работе с регистром (B, D или H) вместо пары регистров с аналогичным именем.

• Смешивание адресов и данных. К типичным ошибкам относятся использование непосредственной адресации вместо прямой адресации или наоборот, смешивание регистров с ячейками памяти, адресуемыми через пары регистров.

• Использование неверных форматов. Типичные ошибки состоят в использовании формата BCD (десятичного) вместо двоичного или наоборот и использование двоичного и шестнадцатеричного кода вместо ASCII.

• Неправильная работа с массивами. Обычная ошибка состоит в выходе за границы массивов.

• Неучет неявных эффектов. К типичным ошибкам относится использование аккумулятора, пары регистров, указателя стека, флагов или ячеек памяти без учета влияния участвующих в работе команд. Большинство ошибок вызываются командами, которые дают непредвиденные, неявные или косвенные результаты.

• Ошибки при задании необходимых начальных условий для отдельных программ или микро-ЭВМ в целом. Большинство программ требует инициализации счетчиков, косвенных адресов, регистров, флагов и ячеек для временного хранения. Микро-ЭВМ в целом требует инициализации всех общих ячеек в ОЗУ (особо отметим косвенные адреса и счетчики).

• Неправильная организация программы. К типичным ошибкам относятся обход или повторение секций инициализации, ошибочное изменение регистров с адресами или счетчиками и потеря промежуточных или окончательных результатов.

Обычным источником ошибок, которые здесь не рассматриваются, является конфликт между программой пользователя и системными программами. Простым примером такого конфликта является попытка сохранять данные для программы пользователя в ячейках памяти системной программы. В этом случае всякий раз, когда выполняется системная программа, изменяются данные, которые нужны для программы пользователя.

Более сложные источники конфликтов связаны с системой прерываний, портами ввода-вывода, стеком и флагами. Системные программы в конечном счете должны эксплуатировать те же самые ресурсы, что и программы пользователя. При этом обычно в системных программах предусматривается сохранение и восстановление программной среды, в которой работают пользовательские программы, но это часто приводит к трудноуловимым или неожиданным последствиям. Сделать такую операционную систему, которая была бы совершенно прозрачной для пользователя – это задача, сравнимая с выработкой правил и законов или сводов о налогах, которые не имели бы лазеек или побочных эффектов.

3.2. ИЗМЕНЕНИЕ ПОРЯДКА ОПЕРАНДОВ НА ОБРАТНЫЙ

Наиболее частыми источниками ошибок являются следующие команды и соглашения:

• Команда MOV R1,R2 пересылает содержимое регистра R2 в регистр R1. Перестановка порядка операндов является простой наиболее общей ошибкой в программах на языке ассебмлера 8080, 8085. Наилучший способ избежать ее – пользоваться системой обозначений операторов,описанной в работе [2].

• Предполагается, что при записи в память 16-разрядных адресов и элементов данных первыми (т. е. в меньших адресах) запоминаются младшие по значению байты. Нарушение этого соглашения становится источником ошибок в командах, которые загружают или запоминают пары регистров или используют стек.

• Команда CMP REG вычитает свой операнд из аккумулятора, а не наоборот. Подобным же образом CPI DATA вычитает значение DATA из аккумулятора.

Примеры

1. **MOV A,B**

Эта команда загружает аккумулятор из регистра B. Так как она не изменяет B, то эта команда действует как *скопировать B в A*.

2. **MOV M,A**

Эта команда запоминает аккумулятор в памяти по адресу, содержащемуся в регистрах H и L. Так как она не изменяет аккумулятор, то эта команда действует как *скопировать A в память по адресу в регистрах H и L*.

3. **LDA 2040H**

Адрес 2040_{16} занимает два байта в памяти программы непосредственно за кодом операции, причем 40_{16} – первый байт, а 20_{16} – второй. Особенно важ-

но помнить этот порядок в том случае, если адрес записывается или изменяется на уровне объектного кода во время отладки.

4. PUSH H

Эта команда запоминает регистры H и L в памяти по адресам непосредственно ниже начального содержимого указателя стека (т. е. по адресам S – 1 и S – 2, если S – начальное содержимое указателя стека). Регистр H запоминается по адресу S –1, а L – по адресу S – 2, т. е. в обычном перевернутом формате.

5. LHLD 2050H

Эта команда загружает регистр L из памяти с адресом 2050_{16}, а H – из 2051_{16}.

6. LHLD 3600H

Эта команда запоминает регистр L в памяти с адресом 3600_{16}, а H – с адресом 3601_{16}.

7. CMP B

Эта команда устанавливает флаги так же, как если бы регистр вычитался из аккумулятора.

8. CPI 25H

Эта команда устанавливает флаги так же, как если бы число 25_{16} вычиталось из аккумулятора.

3.3. НЕПРАВИЛЬНОЕ ИСПОЛЬЗОВАНИЕ ФЛАГОВ

Разные команды микропроцессоров 8080, 8085 по-разному влияют на флаги. При этом общих правил очень мало, и даже команды с похожим смыслом могут выполняться неодинаково. Далее приведены случаи, которые требуют особого внимания.

• Команды пересылки данных, такие как MOV, MVI, LDA, STA, LDAX, LXI, LHLD, SHLD, XCHG и XTHL не влияют ни на какие флаги. Поэтому для установки флагов необходимо использовать лишние арифметические или логические команды (такие, как ANA A, DCR, INR или ORA A).

• Флаги переноса после команд CMP, CPI, SBB, SBI, SUB или SUI указывают на заем. Однако если программист выполняет вычитание с помощью сложения с вычитаемым, представленным в виде дополнения до двух или десяти, флаг переноса является инвертированным заемом, т. е. флаг переноса очищается, если для 8-разрядного беззнакового вычитания может потребоваться заем.

• После сравнения (CMP или CPI) флаг нуля указывает, равны ли операнды; он устанавливается, если операнды равны, и очищается, если они не равны. Здесь присутствует очевидный источник путаницы: JZ означает, *перейти, если результат равен 0*, т. е. *перейти, если флаг нуля равен 1*. Команда JNZ, само собой разумеется, имеет противоположный смысл.

• Если сравниваются беззнаковые числа, флаг переноса указывает, какое из чисел больше. Команда CMP или CPI устанавливает флаг переноса, когда аккумулятор меньше другого операнда, и очищает его, если аккумулятор больше или равен другому операнду. Обратите внимание на то, что если операнды равны, флаг переноса очищается. Если необходимо другое разделение отношения между операндами (т. е. вы хотите различать *больше чем* и *меньше чем или равен*), то надо или менять операнды местами при вычитании, или же добавлять единицу к другому операнду.

• При сравнении чисел со знаками, если не происходит переполнения по дополнению до двух, флаг знака указывает, какой из операндов больше (см. гл. 1). Команды CMP и CPI устанавливают флаг знака, если аккумулятор меньше другого операнда, и очищают его, если аккумулятор больше другого операнда или равен ему. Заметим, что сравнение равных операндов очищает флаг знака. Как и для беззнаковых чисел при равенстве обработка может выполняться противоположным способом с помощью коррекции любого из операндов или реверсирования вычитания.

• Все логические команды за исключением CMA очищают флаг переноса. Команда ANA A или ORA A фактически является быстрым и простым способом очистки флага переноса без влияния на какие-либо регистры. Команда CMA совсем не влияет на флаги. Эквивалентной командой, не влияющей на флаги, является XRI 0FFH.

• Обычный способ выполнения команд при выполнении условия состоит в обходе этих команд, если условие не удовлетворяется. Например, чтобы увеличить регистр B в случае, если флаг переноса равен 1, используют последовательность команд

```
          JNC   NEXT
          INR   B
NEXT:     NOP
```

Переход осуществляется, если флаг переноса равен 0.

• Шестнадцатиразрядные арифметические команды слабо воздействуют на флаги. Команды INX и DCX совсем не влияют на флаги; DAD влияет только на флаг переноса. Ограниченное влияние на флаги указывает на то, что эти команды предназначены для адресной арифметической обработки, а не для обработки 16-разрядных данных.

• Команды INR и DCR не влияют на флаг переноса. Это позволяет использовать их в качестве счетчиков для циклов многобайтной арифметической обработки (когда флаг переноса необходим для переноса или заема между байтами). Однако INR и DCR влияют на флаги нуля и знака, и это может быть использовано для определения того, был ли перенос или заем.

• Команды сдвига (RAL, RAR, RRC, RLC) влияют только на флаг переноса.

• Арифметические и логические команды специального назначения, такие команды, как ADD A (логический сдвиг аккумулятора влево), ADC A (циклический сдвиг аккумулятора влево), SUB A (очистка аккумулятора) и ANA A или ORA A (проверка аккумулятора), влияют на все флаги.

• Команды PUSH и POP не влияют на флаги, за исключением команды POP PSW, которая изменяет все флаги. Напомним, что слово состояния процессора (PSW) содержит аккумулятор (старший байт) и флаги (младший байт).

Примеры

1. Последовательность команд

```
          LDA   2040H
          JZ    DONE
```

имеет непредсказуемый результат, так как LDA не влияет на флаги. Для того чтобы совершить переход, если ячейка памяти 2040_{16} содержит 0, используйте

```
LDA  2040H
ANA  A          ;ПРОВЕРИТЬ АККУМУЛЯТОР
JZ   DONE
```

Вместо ANA A можно воспользоваться командой ORA A.

2. Последовательность команд

```
MOV  A,E
JP   DEST
```

имеет непредсказуемый результат, так как MOV не влияет на флаги. Любая из данных последовательностей команд вызывает переход, если регистр E положительный:

```
MOV  A,E
ANA  A
JP   DEST
```

или

```
SUB  A
ORA  E
JP   DEST
```

3. Команда CPI 25H устанавливает флаг переноса следующим образом:

флаг переноса = 1, если содержимое регистра A находится между 00 и 24_{16};
флаг переноса = 0, если содержимое регистра A находится между 25_{16} и FF_{16}.

Флаг переноса устанавливается, если регистр A содержит беззнаковое число, меньшее другого операнда, и очищается, если A содержит беззнаковое число, большее другого операнда или равное ему.

Если вы желаете установить флаг переноса в случае, когда аккумулятор содержит 25_{16}, вместо CPI 25H воспользуйтесь командой CPI 26H. В результате будет

```
CPI  25H
JC   LESS       ;ПЕРЕЙТИ, ЕСЛИ (A) МЕНЬШЕ ЧЕМ 25
```

или

```
CPI  26H
JC   LESSEQ     ;ПЕРЕЙТИ, ЕСЛИ (A) МЕНЬШЕ ИЛИ РАВНО 25
```

4. Последовательность команд

```
RAL
JP   DONE
```

имеет непредсказуемый результат, так как RAL не влияет на флаг знака. Правильной будет последовательность (выполняющая циклический сдвиг, который влияет на флаги)

```
ADC  A          ;СДВИНУТЬ ЦИКЛИЧЕСКИ, УСТАНОВИВ ФЛАГИ
JP   DONE
```

Конечно, можно также использовать более медленный вариант

```
RAL
RAL
JC   DONE
```

5. Последовательность команд

```
INR  B
JC   OVRFLW
```

имеет непредсказуемый результат, так как INR не влияет на флаг переноса. Правильной будет последовательность

```
INR  B
JZ   OVRFLW
```

так как INR влияет на флаг нуля.

6. Последовательность команд

```
DCR  B
JC   OVRFLW
```

имеет непредсказуемый результат, так как DCR не влияет на флаг переноса. Если B не может содержать число, большее чем 80_{16} (беззнаковое), используйте

```
DCR  B
JM   OVRFLW
```

так как DCR не влияет на флаг знака. Заметим, однако, что если B первоначально содержит 81_{16}, то произойдет ошибочный переход.

Более длинная, но одновременно и более общая последовательность команд выглядит следующим образом:

```
INR  B          ;ПРОВЕРИТЬ РЕГИСТР B
DCR  B
JZ   OVRFLW     ;ПЕРЕЙТИ, ЕСЛИ B СОДЕРЖИТ НУЛЬ
DCR  B
```

Заметим, что если в программе осуществляется переход по адресу OVRFLW, то регистр B будет содержать 0 (не FF_{16}).

7. Последовательность команд

```
DCX  B
JNZ  LOOP
```

имеет непредсказуемый результат, так как DCX не влияет ни на один флаг. Правильная последовательность для уменьшения и проверки 16-разрядного счетчика в паре регистров B будет

```
DCX  B
MOV  A,C        ;ПРОВЕРИТЬ, ЕСТЬ ЛИ В ПАРЕ РЕГИСТРОВ
ORA  B          ; BC КАКИЕ-ЛИБО РАЗРЯДЫ, РАВНЫЕ 1
JNZ  LOOP       ;BC НЕ МОЖЕТ БЫТЬ НУЛЕМ, ЕСЛИ
                ; ХОТЯ БЫ ОДИН РАЗРЯД РАВЕН 1
```

Эта последовательность влияет на аккумулятор и все флаги, включая флаг переноса (который очищает команда ORA).

8. Команда ANA A и ORA A очищают флаг переноса без влияния на какие-либо регистры. Для того чтобы очистить флаг переноса без изменения остальных флагов, используйте последовательность

```
STC             ;СНАЧАЛА УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА
CMC             ;ЗАТЕМ ОЧИСТИТЬ, ИНВЕРТИРОВАВ ЕГО
```

9. Команды SUB A и XRA A очищают аккумулятор, флаг переноса и флаг знака (и устанавливают флаг нуля). Для того чтобы очистить аккумулятор без влияния на флаги, используйте MVI A,0.

10. Последовательность команд

```
LXI  D,-VAL16
DAD  D
JZ   BNDRY
```

имеет непредсказуемый результат, так как DAD не влияет на флаг нуля. Для того чтобы вызвать переход, если H и L содержат VAL16, проверьте H и L с помощью следующих команд:

```
LXI  D,-VAL16
DAD  D
MOV  A,H         ;ПРОВЕРИТЬ H И L НА НУЛЬ
ORA  L
JZ   BNDRY
```

3.4. СМЕШИВАНИЕ РЕГИСТРОВ И ПАР РЕГИСТРОВ

Запомните следующие правила:

- MOV, LDAX, STAX и MVI предназначены для работы с одиночными регистрами;
- LHLD, LXI, POP, PUSH, XCHG и XTHL предназначены для работы с парами регистров;
- регистр M ссылается на байт памяти, расположенный по адресу, содержащемуся в регистрах H и L; он не относится к самим регистрам H и L.

Типичные ошибки состоят в использовании MOV A,H вместо MOV A,M, или наоборот, LDAX вместо MOV или MVI вместо LXI, или наоборот. Применение пары регистров для хранения адреса означает, что некоторые команды будут необычными. Например MOV L,M может загрузить регистр L из адреса, содержащегося в H и L; после этого H и L будут содержать один байт адреса (регистр H) и один байт данных (регистр L). Это хотя и допустимо, но редко бывает полезным.

Примеры

1. MOV A,H

Эта команда пересылает регистр H в аккумулятор. Она не изменяет регистр H или какую-нибудь ячейку памяти.

2. LDAX B

Эта команда загружает аккумулятор из памяти с адресом, содержащимся в регистрах B и C. Она не влияет ни на регистр B, ни на регистр C.

3. MVI H,0

Эта команда помещает 0 в регистр H. Она не оказывает влияния на память.

4. MOV M,A

Эта команда запоминает аккумулятор в ячейке памяти, адресуемой регистрами H и L. Она не влияет ни на H, ни на L. Следующая последовательность команд загружает в H и L косвенный адрес:

```
MOV  E,M         ;ВЗЯТЬ МЛАДШИЙ БАЙТ КОСВЕННОГО АДРЕСА
INX  H
MOV  D,M         ;ВЗЯТЬ СТАРШИЙ БАЙТ КОСВЕННОГО АДРЕСА
XCHG             ;ПОМЕСТИТЬ КОСВЕННЫЙ АДРЕС В HL
```

Для того чтобы использовать только один временный регистр хранения, следует загружать старший байт непосредственно в H следующим образом:

```
MOV  A,M      ;ВЗЯТЬ МЛАДШИЙ БАЙТ КОСВЕННОГО АДРЕСА
INX  H
MOV  H,M      ;ВЗЯТЬ СТАРШИЙ БАЙТ КОСВЕННОГО АДРЕСА
MOV  L,A      ;ПЕРЕСЛАТЬ МЛАДШИЙ БАЙТ АДРЕСА В L
```

Эта последовательность выполняется за то же число временных циклов, что и предыдущая, однако использует для временного хранения регистр A вместо пары регистров D и E.

5. **LXI H,2050H**

Эта команда загружает 2050_{16} в пару регистров HL (20_{16} в H и 50_{16} в L).

6. **ADD M**

Эта команда добавляет к аккумулятору байт памяти, адресуемый регистрами H и L. Она не влияет ни на H, ни на L.

7. **DAD H**

Эта команда добавляет пару регистров H к себе самой, сдвигая, таким образом, логически H и L влево на один разряд. Она не влияет на аккумулятор и не выбирает данные из памяти.

3.5. СМЕШИВАНИЕ АДРЕСОВ И ДАННЫХ

Запомните следующие правила:

- команды LDA, STA, LHLD, JMP и CALL требуют в качестве операнда адрес;
- команды MVI и LXI требуют данные.

Терминология адресации для переходов и вызовов подпрограмм несколько запутана. Хотя и считается, что в этих командах используется прямая адресация, однако они обрабатывают свои операнды скорее как LXI, чем как LHLD. Например, JMP 2040H загружает 2040_{16} в счетчик команд почти так же, как LXI H,2040H загружает 2040_{16} в пару регистров H, в то время как LHLD 2040H загружает в пару регистров H содержимое ячеек памяти 2040_{16} и 2041_{16}.

Примеры

1. Команда MVI A,40H загружает число 40_{16} в аккумулятор. Команда LDA 40H загружает в аккумулятор содержимое ячейки памяти 0040_{16}.

2. Команда LXI H,0C00H загружает в регистры H и L число $0C00_{16}$ ($0C_{16}$ в H и 00_{16} в L). Команда LHLD 0C00H загружает в пару регистров H содержимое ячеек памяти $0C00_{16}$ и $0C01_{16}$ (содержимое $0C00_{16}$ в L и содержимое $0C01_{16}$ в H).

Перепутывание адресов и их содержимого является распространенной ошибкой при работе со структурами данных. Например, очередь задач, которые должны выполняться элементом тестового оборудования, должна содержать для каждой задачи некоторый блок информации. Этот блок мог бы содержать:

стартовый адрес тестовой программы,
время в секундах, за которое должен прогоняться тест,
адрес, по которому должен быть сохранен результат,

верхнее и нижнее пороговые значения, с которыми должны сравниваться результаты,

базовый адрес следующего блока в очереди.

Таким образом, этот блок содержит данные, прямые адреса и косвенные адреса. Типичные ошибки, которые мог бы сделать программист, следующие:

передача управления по адресу ячеек памяти, содержащих стартовый адрес тестовой программы, а не по действительному стартовому адресу;

запоминание результата в блоке, а не по адресу, содержащемуся в блоке;

использование пороговых значений в качестве адресов, а не данных;

предположение, что следующий блок начинается в текущем блоке, а не по базовому адресу, данному в текущем блоке.

Другим распространенным источником ошибок являются таблицы переходов. Возможны следующие альтернативные подходы:

сформировать таблицу команд переходов и передавать управление нужному элементу (например, третьей команде перехода);

сформировать таблицу адресов назначения и передавать управление по содержимому нужного элемента (например, по адресу в третьем элементе).

Само собой разумеется, если команды переходов используются в качестве адресов, или наоборот, это приведет к ошибкам.

3.6. ОШИБКИ ФОРМАТА

Запомните следующие правила для стандартного ассемблера 8080, 8085:

• наличие H в конце числа указывает на то, что число шестнадцатеричное, а B — что число двоичное;

• по умолчанию число считается десятичным, т. е. в ассемблере считается, что все числа являются десятичными, если они не отмечены как-либо иначе;

• шестнадцатеричному числу, начинающемуся с буквенной цифры (A, B, C, D, E или F), должен предшествовать 0 (например, 0CFH вместо CFH), для того чтобы ассемблер правильно интерпретировал это число. Разумеется, 0 в начале не влияет на значение числа;

• все арифметические и логические команды, за исключением DAA, которая корректирует результат 8-разрядного двоичного сложения в соответствующее значение в коде BCD, оперируют с двоичными числами.

Остерегайтесь следующих распространенных ошибок:

• пропуска H у шестнадцатеричного операнда. В ассемблере число будет восприниматься как десятичное, если оно не содержит буквенных цифр, и как имя, если оно начинается с буквы. Ассемблер укажет на ошибку только в том случае, если он не сможет интерпретировать операнд как десятичное число или как имя;

• пропуска B после двоичного операнда. В ассемблере он будет приниматься за десятичный;

• перепутывания десятичного (BCD) представления с вдоичным представлением. Помните, что 10 не является целой степенью двух, так что для чисел больше 9 двоичное и BCD представления неодинаковы. Константы BCD должны обозначаться как шестнадцатеричные числа, а не как десятичные;

• перепутывания двоичного и десятичного представления с представлением в коде ASCII. Устройство ввода ASCII дает символы ASCII, а любое устройство вывода ASCII реагирует на символы ASCII.

Примеры

1. LDA 2000

Эта команда загружает аккумулятор из адреса памяти 2000_{10} ($07D0_{16}$), а не из адреса 2000_{16}. Ассемблер не укажет на ошибку, так как 2000 является правильным десятичным числом.

2. ANI 00000011

Эта команда выполняет логическую операцию И аккумулятора с десятичным числом 11 (1011_2), а не с двоичным числом 11 (3_{10}). Ассемблер не укажет на ошибку, так как 00000011 является правильным десятичным числом, несмотря на необычную форму.

3. ADI 40

Эта команда добавляет число 40_{10} к аккумулятору. Заметим, что 40_{10} – это не то же самое, что 40 в коде BCD, которое будет равно 40_{16}, в то время как 40_{10} равняется 28_{16}. Ассемблер не укажет на ошибку, так как 40 является правильным десятичным числом.

4. MVI A,3

Эта команда загружает в аккумулятор число 3. Если теперь это значение посылается на устройство вывода ASCII, то устройство будет реагировать на него как на символ ETX (03_{16}), а не на символ 3 (33_{16}). Правильный вариант

```
MVI  A,'3'      ;ВЗЯТЬ 3 В КОДЕ ASCII
```

5. Если ячейка памяти 2040_{16} содержит одну десятичную цифру, то с помощью последовательности команд

```
LDA  2040H
OUT  DEVCE
```

это число не будет напечатано на устройстве вывода ASCII. Правильной будет последовательность

```
LDA  2040H      ;ВЗЯТЬ ДЕСЯТИЧНУЮ ЦИФРУ
ADI  '0'        ;ПРЕОБРАЗОВАТЬ В КОД ASCII
OUT  DEVCE
```

6. Если порт ввода INDEV содержит одну десятичную цифру ASCII, то последовательность команд

```
IN   INDEV
STA  2040H
```

не обеспечит запоминания действительной цифры в ячейке памяти 2040_{16}. Вместо этого будет запомнен код ASCII, который является десятичной цифрой плюс 30_{16}. Правильной будет последовательность команд

```
IN   INDEV      ;ВЗЯТЬ ЦИФРУ В КОДЕ ASCII
SUI  '0'        ;ПРЕОБРАЗОВАТЬ В ДЕСЯТИЧНУЮ
STA  2040H
```

Выполнение десятичных арифметических операций на процессорах 8080, 8085 вызывает затруднения, так как DAA выполняется только после команд 8-разрядного сложения (ACI, ADC, ADD и ADI). Она не выполняется так, как положено, после вычитания, увеличения, уменьшения или двойного сложения. В гл. 6 приведены подпрограммы для десятичных арифметических операций. Так как DAA не выполняется правильно после DCR или INR, то

для увеличения и уменьшения на 1 необходимы такие последовательности команд:

- Добавить 1 к аккумулятору в десятичном виде

```
ADI  1
DAA
```

- Вычесть 1 из аккумулятора в десятичном виде

```
ADI  99H
DAA
```

Здесь флаг переноса является инвертированным заемом.

3.7. НЕПРАВИЛЬНАЯ РАБОТА С МАССИВАМИ

Наиболее распространенной ошибкой при работе с массивами является выполнение лишней итерации или прекращение итераций раньше времени. Напомним, что ячейка памяти от BASE до BASE + N содержит N + 1, а не N байт. Легко забыть про последнюю запись массива или пропустить первую. С другой стороны, если есть N записей, то они занимают ячейки памяти от BASE до BASE + N – 1; из-за этого может легко произойти выход за пределы массива.

3.8. НЕЯВНЫЕ ЭФФЕКТЫ

Необходимо помнить о существовании следующих неявных эффектов:

- очистке флага переноса всеми логическими командами за исключением CMA;
- изменении обеих половин пары регистров такими командами, как LXI, LHLD, INX, DCX и XCHG;
- использовании адреса памяти, на единицу большего адреса, определенного в командах LHLD и SHLD;
- изменении указателя стека командами POP, PUSH, CALL, RET и RST;
- сохранении в стеке адреса возврата командами CALL и RST;
- использовании адреса, содержащегося в паре регистров H и L, командами, обращающимися к регистру M;
- изменении H и L командой DAD;
- использовании аккумулятора командами IN, LDAX, OUT, RIM, SIM и STAX; команды RIM и SIM предусмотрены только в микропроцессоре 8085.

Примеры

1. ANI 00001111B

Эта команда очищает флаг переноса, а также выполняет логическую операцию.

2. INX H

Эта команда добавляет 1 к паре регистров H, а не к H и L в отдельности. Фактически INX H добавляет 1 к регистру L, а затем добавляет перенос, если он есть, к регистру H. Напомним, что INR и DCR – 8-разрядные команды, в то время как INX и DCX – 16-разрядные.

3. LXI D,2050H

Эта команда изменяет как регистр D (на 20_{16}), так и регистр E (на 50_{16}).

4. LHLD 16EFH

Эта команда использует адреса памяти 16 EF_{16} и 16 $F0_{16}$. Она загружает регистр L из адреса 16 EF_{16}, а регистр H – из адреса 16 $F0_{16}$.

5. PUSH H

Эта команда не только сохраняет регистры H и L в памяти, но также и уменьшает указатель стека на 2.

6 RET

Эта команда не только загружает счетчик команд из двух верхних ячеек стека, но и увеличивает указатель стека на 2.

7. INR M

Эта команда добавляет 1 к содержимому ячейки памяти, адресуемой парой регистров H. Отметим, что это не то же самое, что INR H (которая добавляет 1 к регистру H), INR L (которая добавляет 1 к регистру L) или INX H (которая добавляет 1 к паре регистров H).

8. DAD D

Эта команда добавляет 16-разрядное число, находящееся в паре регистров D, к 16-разрядному числу в паре регистров H и сохраняет результат в паре регистров H. Старое содержимое пары регистров H при этом теряется, а пара регистров D не изменяется.

9. IN 20H

Эта команда загружает аккумулятор из порта 20_{16}. Старое содержимое аккумулятора теряется.

10. RIM

Эта команда (только в 8085) загружает аккумулятор масками прерываний. Старое содержимое аккумулятора теряется.

3.9. ОШИБКИ ИНИЦИАЛИЗАЦИИ

Для системы микро-ЭВМ в целом и для отдельных программ программы инициализации должны осуществлять следующее:

- загружать все ячейки ОЗУ начальными значениями. Сюда входят косвенные адреса и другие ячейки для временного хранения. Неправомерно считать, что ячейки памяти содержат 0 только потому, что они не использовались;
- загружать все регистры и флаги начальными значениями. Сброс инициализирует только систему прерываний (запрещая их). Программа начального запуска должна, например, инициализировать указатель стека с использованием команды LXI SP или последовательности команд LHLD, SPHL;
- инициализировать все счетчики и косвенные адреса. Особо отметим пары регистров, которые используются в качестве адресных регистров. Напомним, что пара регистров H должна быть загружена адресом до использования команд, которые ссылаются на регистр M.

3.10. НЕПРАВИЛЬНАЯ ОРГАНИЗАЦИЯ ПРОГРАММЫ

Следующие ошибки являются наиболее распространенными.

- Случайная переинициализация регистра, пары регистров, флага, ячейки памяти, счетчика или косвенного адреса. Необходимо быть уверенным в том, что переходы не приведут к повторению команд инициализации.
- Ошибки при обновлении счетчика или косвенного адреса. Трудность здесь состоит в том, что возможен путь, при котором обходятся команды обновления или же изменяются значения перед выполнением этих команд.

• Несохранение результатов регистров. Ничего не стоит вычислить результат, а затем загрузить в аккумулятор что-нибудь другое. Выявление подобных ошибок – это полная разочарований работа, отнимающая много времени, так как все команды вычисления результата верны, а сам результат теряется. Например, команда перехода может передавать управление команде, которая изменяет результат.

• Отсутствие обхода команд, которые не должны выполняться в данной ветви программы. Напомним, что ЭВМ выполняет команды последовательно до тех пор, пока не будет указано на изменение порядка работы. Таким образом, ЭВМ может ошибочно приступить к выполнению той секции программы, которую она должна была бы выполнять только при переходе. Поэтому необходима команда безусловного перехода, вызывающая обход той секции, которая не должна выполняться.

3.11. РАСПОЗНАВАНИЕ ОШИБОК АССЕМБЛЕРОМ

Большинство ассемблеров немедленно распознает наиболее распространенные ошибки, такие как:

• неопределенный код операции (обычно это неправильное написание или же отсутствие двоеточия после метки);

• неопределенное имя (часто это неправильное написание или отсутствие определения имени);

• недопустимый символ (например, 2 в двоичном числе или В в десятичном числе);

• неверный формат (например, неправильный разделитель или ошибочные операнды);

• неправильное значение (обычно это число, которое слишком велико для 8 или 16 разрядов);

• отсутствующий операнд;

• двойное определение (одному и тому же имени присваиваются два различных значения);

• недопустимая метка (например, метка, предписанная псевдооперации, не допускающей метки);

• отсутствие метки (например, при псевдооперации EQU, для которой требуется метка).

Эти ошибки неприятны, но они легко исправимы. Единственная трудность возникает тогда, когда ошибка (такая, как отсутствие точки с запятой у строки с комментарием) приводит ассемблер в "замешательство", результатом чего является ряд бессмысленных сообщений об ошибках.

Существует, однако, много простых ошибок, которые ассемблеры не могут распознать. Программисту следует иметь в виду, что его программа может содержать такие ошибки, даже если ассемблер и не сообщил о них. Типичны следующие примеры.

• Пропущенные строки. Очевидно, что ассемблер не может сообщить о том, что какая-то строка была полностью пропущена, если только она не содержит метку или определение, которые используются в другом месте. Легче всего пропустить повторяющиеся или кажущиеся лишними строки. Типичными повторениями являются последовательности сдвигов, переходов, увели-

чений или уменьшений регистров. К командам, которые часто кажутся ненужными, относятся ANA A, DCX H, INX H, ORA A и SUB A.

• Пропущенные определения. Ассемблер не может выдать сообщение о том, подразумевается ли шестнадцатеричный или двоичный операнд, если только пропуск определения не приводит к запрещенной цифре (как С в двоичном числе). Иначе говоря, в ассемблере все числа считаются десятичными. Проблемы часто возникают для шестнадцатеричных чисел, которые не содержат буквенных цифр (например, таких чисел как 44 или 2050) и для двоичных чисел (например, 00000110).

• Ошибки в написании, когда запись сама по себе допустима. Типичными примерами являются ввод команд AND или ADC вместо ADD, ORA вместо XRA, D вместо B (в качестве регистра или пары регистров) или D вместо E (в качестве регистра). В ассемблере не предусмотрены средства распознавания такой ошибки – он реагирует только на недопустимые записи. Подобные "допустимые" ошибки в написании часто встречаются в тех случаях, когда программист использует такие имена, как XXX и XXXX, L121 и L112 или VAR1I и VARII.

• Обозначение команд как комментариев. Если точка с запятой ставится в начале командной строки, ассемблером эта строка воспринимается как комментарий. Такая ошибка может сбить с толку, так как эта строка появляется в листинге, но не ассемблируется в код команды.

Иногда из-за ввода совершенно неправильных команд ассемблер может оказаться в тупиковой ситуации. Ассемблер может принять их просто потому, что его разработчик никак не мог предусмотреть подобные ошибки. При этом результат может быть непредсказуемым. Вот некоторые из случаев, когда ассемблер 8080, 8085 может работать неправильно.

• Если в команде, которая работает с парой регистров, задается одиночный регистр. В некоторых ассемблерах допустимы команды типа INX L, DAD C или LXT E,2040H. При этом они могут создать бессмысленный объектный код без какого-либо указания на ошибку.

• Если вводится неправильная цифра, такая как X в десятичном или шестнадцатеричном числе или 7 в двоичном числе. В некоторых ассемблерах такой неправильной цифре может быть присвоено произвольное значение.

Если вводится неправильный операнд, такой как 8 в RST, PSW в LXI или SP в PUSH или POP. Некоторые ассемблеры могут принять это и генерировать бессмысленный код.

В ассемблере могут распознаваться только такие ошибки, которые предусмотрел его разработчик. Программисты же часто способны делать ошибки, которые разработчик не мог и вообразить, подобно тому, как водители автомобилей часто способны создавать дорожные происшествия, до которых службы дорожной безопасности не способны даже додуматься. Заметим, что ошибки, которые не распознаются ассемблером, могут быть найдены только при проверке программ вручную строчка за строчкой.

3.12. РАСПРОСТРАНЕННЫЕ ОШИБКИ В ДРАЙВЕРАХ ВВОДА-ВЫВОДА

Так как большинство ошибок в драйверах ввода-вывода связано как с аппаратурным, так и с программным обеспечением, они трудно поддаются классификации. Приведем некоторые возможные случаи.

• Смешивание портов ввода и вывода. В большинстве систем порт ввода 20_{16} и порт вывода 20_{16} различаются. Даже в тех случаях, когда физически функции двух портов осуществляются одним и тем же портом, все же невозможно считать в процессор выводимые данные, если только контроллер, обслуживающий этот порт, не сохраняет данные в регистре-защелке или не буферирует их.

• Попытка выполнить операции, которые физически невозможны. Так, чтение данных с устройства вывода (например, с экрана дисплея) или выдача данных на устройство ввода (например, клавиатуру) не вызывают никакого действия. Однако такие случайные ошибки, как использование неправильного номера порта, ассемблером не распознаются; порт, в конце концов, существует, и в ассемблере не предусмотрены средства для того, чтобы узнать о невозможности выполнения определенной операции. Подобным же образом, в программе может быть сделана попытка сохранить данные по несуществующему адресу или в ПЗУ.

• Упущение из вида неявных эффектов аппаратуры. Передаваемые в порт или из порта данные иногда автоматически изменяют линии состояния (как, например, в режиме работы 1 для параллельного интерфейса 8255). Даже чтение из порта или запись в порт во время отладки программы могут изменить линии состояния. В тех случаях, когда используется ввод-вывод, отраженный на память, будьте особенно внимательны при выполнении команд сравнения; эти команды предназначены для чтения данных из памяти с заданным адресом без изменения при этом каких-либо регистров. Подобным образом команды, такие как DCR M и INR M, обеспечивают как чтение из памяти, так и запись в память по адресу. Автоматические операции с портом могут сократить и упростить программы, однако, используя их, надо точно знать, как и когда они осуществляются.

• Чтение или запись без проверки состояния. Многие устройства могут принимать или передавать данные только тогда, когда линия состояния указывает на их готовность. Передача данных к ним или от них при отсутствии готовности может давать непредсказуемые результаты.

• Игнорирование различия между вводом и выводом. Напомним, что устройство ввода начинает работать обычно с состояния *не готово*, так как данных еще нет, хотя ЭВМ и готова их получать. С другой стороны, устройство вывода начинает работать обычно с состояния *готово*, т. е. оно может принимать данные, но ЭВМ обычно нечего послать. Во многих ситуациях (особенно, когда используются программируемые устройства ввода-вывода 8155, 8156 или 8255), нужно послать нулевой символ (00_{16}) (или что-нибудь подобное, не вызывающее действия) во все порты вывода просто для того, чтобы изменить состояния с *готово* на *не готово*.

• Ошибка при сохранении копии выводимых данных. Напомним, что не всегда возможно считать в процессор данные из порта вывода. Если они потребуются вам позднее для повторения передачи, которая была неправильно осуществлена, или часть их надо изменить (например, включить или выключить одну из нескольких индикаторных ламп, присоединенных к одному и тому же порту вывода), или сохранить данные, характеризующие состояние на момент прерывания (например, текущий уровень прерывания), следует

сохранить их копию в памяти. Эта копия должна обновляться каждый раз при изменении действительных данных.

• Чтение данных до того, как они стабилизируются, или во время их изменения. Совершенно необходимо точно знать тот момент, когда данные установились и их можно читать. Например, если переключатели имеют дребезг, то программа до начала обработки может получить сигналы о двух крайних состояниях. В случае, если кнопки имеют дребезг, лучше, чтобы обработка начиналась при их отжатии, а не нажатии; это заставляет оператора отпускать кнопку, а не держать ее в нажатом состоянии. В случае устойчивых данных (как, например, при последовательном вводе-выводе), момент получения данных необходимо центрировать (т. е. читать данные, соответствующие центру импульса, а не краям, где значение может меняться).

• Отсутствие изменения полярности данных, которые передаются к устройству или от устройства, работающего с отрицательной логикой. Во многих простых устройствах ввода-вывода, таких как выключатели или устройства индикации, используется отрицательная логика; при этом логический 0 означает, что выключатель нажат или же индикатор светится. В обычных наборных дисках, как и в большинстве кодирующих устройств, также часто при выдаче данных используется отрицательная логика. Выход в этом случае простой: после чтения или перед выдачей данных инвертировать их, используя команду СМА.

• Смешивание действительных портов ввода-вывода с внутренними регистрами интегральных схем ввода-вывода. Программируемые устройства ввода-вывода, такие как 8155, 8156, 8255 и 8279, содержат обычно управляющие или командные регистры (которые определяют, как работать устройству) и регистры состояния (которые отражают текущее состояние устройства или передачи). Эти регистры находятся внутри интегральной схемы и не связаны с периферийными устройствами. Данные, передаваемые на эти регистры или с них, — совершенно не те данные, которые передаются в действительный порт ввода-вывода или из него.

• Неправильное использование двунаправленных портов. Многие устройства, такие как 8155, 8156, 8255, 8355 и 8755, снабжены двунаправленными портами ввода-вывода, которые могут использоваться как для ввода, так и для вывода. Обычно во избежание начальной неопределенности ЭВМ при сбросе делает эти порты портами ввода, так что если это необходимо, программа должна изменять их на порты вывода. Будьте особенно осторожны с командами чтения разрядов или портов, назначенных для вывода, а также с командами записи в разряды или порты, назначенные для ввода. Единственный способ избежать неприятностей — внимательно изучать документацию на каждое конкретное устройство.

• Отсутствие очистки состояния после выполнения команды ввода-вывода. Как только процессор считает данные из порта или запишет данные в какой-либо порт, этот порт должен вернуться в *состояние не готов*. Некоторые устройства ввода-вывода автоматически изменяют состояние своих портов после команд ввода или вывода, а другие либо вообще не изменяют, либо автоматически изменяют состояние, но только после ввода. Если оставить состояние установленным, то результатом может быть бесконечный цикл или же неустойчивая операция.

3.13. РАСПРОСТРАНЕННЫЕ ОШИБКИ В ПРОГРАММАХ ОБСЛУЖИВАНИЯ ПРЕРЫВАНИЙ

Многие ошибки, связанные с прерываниями, зависят как от аппаратного, так и программного обеспечения. Самыми распространенными ошибками являются следующие.

- Отсутствие разрешения прерываний. Процессоры 8080 и 8085 после появления сигнала прерывания автоматически запрещают прерывания и не разрешают их вновь, пока не будет выполнена команда EI.
- Отсутствие сохранения регистров. Ни 8080, ни 8085 не сохраняют автоматически ни одного регистра, кроме счетчика команд. Любые регистры, которые использует программа обслуживания прерываний, должны с помощью соответствующих команд сохраняться в стеке.
- Сохранение или восстановление регистров в неправильном порядке. Регистры должны восстанавливаться в порядке, обратном тому, в котором они записывались в стек.
- Разрешение прерываний до инициализации приоритетов и других параметров системы прерываний.
- Неучет того, что реакция на прерывание включает сохранение счетчика команд в вершине стека. Таким образом, адрес возврата должен находиться выше всех остальных данных, хранящихся в стеке.
- Отсутствие запрещения прерываний во время многобайтных передач или выполнения последовательности команд, которая не должна прерываться. В особенности следите за частичным обновлением данных (таких как время), которые могут использоваться в программе обслуживания прерываний.
- Отсутствие разрешения прерываний после последовательности команд, которая должна выполняться без прерываний. Проблема состоит здесь в том, что нельзя разрешать прерывания, если они не были разрешены ранее. В микропроцессоре 8080 это вызывает определенные трудности, так как нельзя прочитать состояние системы прерываний, как это делается в 8085. Решение заключается в необходимости хранить в памяти флаг, указывающий на разрешение или запрещение в данный момент прерываний. Разумеется, этот флаг должен обновляться после каждой команды DI и перед каждой командой EI [15].
- Отсутствие очистки сигнала, вызвавшего прерывание. Обслуживающая программа должна сбрасывать прерывание, даже если нет необходимости в операциях ввода-вывода. Например, даже если процессор не имеет данных для передачи устройству вывода, вызвавшему прерывание, он все же должен или очистить сигнал данного прерывания, или запретить его. В противном случае процессор попадает в бесконечный цикл. Это относится и к часам реального времени: хотя они и не требуют обычно никакого другого обслуживания, кроме обновления времени, обслуживающая программа все же должна очищать сигнал прерывания. Эта очистка может заключаться в чтении регистра программируемого таймера.
- Ошибка в общении с основной программой. Основная программа не может иметь информации о том, что прерывание уже обработано до тех пор, пока не будет ей об этом сообщено. Обычный способ информирования основной программы заключается в том, чтобы заставить обслуживающую программу изменить флаг. По значению этого флага программа может получить

информацию о том, что выполнение программы уже закончено. Эта процедура сравнима с практикой, принятой в почтовых отделениях: начальник отделения поднимает флаг, указывающий на наличие корреспонденции, которую необходимо выбрать, а почтальон после того, как заберет корреспонденцию, опускает этот флаг. При такой простой процедуре основной программой может проверяться этот флаг так часто, как нужно для того, чтобы не пропустить факта изменения его значения. При этом программист, разумеется, всегда обязан обеспечивать буфер, достаточный для того, чтобы в нем можно было хранить много элементов данных.

• Отсутствие сохранения и восстановления приоритетов. Приоритет прерывания часто содержится в регистре, допускающем только запись в него, или в какой-либо ячейке памяти. Этот приоритет должен точно так же храниться в обычном регистре и правильно восстанавливаться в конце обслуживающей программы. Если в регистр приоритета можно только записывать (как, например, в устройстве управления прерываниями 8214), то каждая программа, изменяющая его, должна сохранять копию его содержимого в памяти.

Микропроцессор 8085 имеет более сложную систему прерываний, чем 8080. Эти дополнительные особенности микропроцессора 8085 приводят и к новым ошибкам, к которым относятся:

• Отсутствие разрешения прерываний от дополнительных аппаратурных входов (RST 5.5, RST 6.5 и RST 7.5), которое выполняется с помощью очистки разрядов масок в регистре I. Необходимо быть внимательным: эти разряды являются масками, а не разрешениями прерываний. Для разрешения прерываний эти разряды масок должны быть очищены, а для запрещения – установлены. Таким образом, разряды масок имеют полярность, противоположную полярности флага разрешения прерываний. Поэтому, если даже разряды масок и были очищены, все равно, для того, чтобы была возможность распознать маскируемые прерывания, флаг разрешения прерываний должен быть установлен (с помощью команды EI). В микропроцессоре 8085 предусмотрено также немаскируемое аппаратурное прерывание (TRAP или RST 4.5), которое часто используется для индикации уменьшения напряжения ниже допустимого предела.

• Неправильное использование разрядов разрешения прерываний в командах SIM. Команда SIM имеет две различные функции: установка или очистка масок прерываний и выдача значения на линию SOD (данные последовательного вывода). Разрешение установки масок (разряд 3) и разрешение последовательного вывода (разряд 6) позволяют программисту осуществлять одну функцию без влияния на другие, т. е., установив разряд 3 и очистив разряд 6, программист может изменить маски прерываний без изменения последовательного вывода. Аналогично, очистив разряд 3 и установив разряд 6, программист может изменить последовательный вывод, не изменяя маски прерываний. Эти разряды разрешения оказались необходимы, так как разработчики 8085 решили использовать один регистр для двух несвязанных функций.

Заметим также, что в 8085 регистр I, записываемый командой SIM, отличается от регистра I, читаемого командой RIM.

ВВЕДЕНИЕ В ПРОГРАММНЫЙ РАЗДЕЛ

Программный раздел содержит наборы подпрограмм на языке ассемблера для микропроцессоров 8080 и 8085. Каждой подпрограмме предпосланы введение и комментарии; за каждой подпрограммой следует по крайней мере один пример ее использования. Введение содержит следующую информацию: назначение подпрограммы, процедуру ее выполнения, используемые регистры, время выполнения, размер памяти, необходимой для программы и ее данных, а также специальные случаи, входные и выходные условия.

Каждая подпрограмма сделана настолько общей, насколько это возможно. Для программ ввода-вывода и обслуживания прерываний, описанных в гл. 10 и 11, это требование выполнить трудно, так как на практике эти подпрограммы всегда зависят от ЭВМ. В этих случаях зависимость от ЭВМ была ограничена таким образом, чтобы она появлялась только для общих управляющих секций ввода-вывода и диспетчеров прерываний. При этом конкретные примеры показаны для ЭВМ, ориентированных на распространенную операционную систему CP/M, однако общие принципы применимы и для других ЭВМ, базирующихся на микропроцессорах 8080 и 8085.

Во всех подпрограммах был использован следующий метод передачи параметров:

1. Первый 8-разрядный параметр передается в аккумуляторе, второй 8-разрядный параметр – в регистре B, а третий – в регистре C.

2. Первый 16-разрядный параметр передается в регистрах H и L, при этом старший байт – в регистре H. Второй 16-разрядный параметр передается в регистрах D и E со старшим байтом в D.

3. Большее число параметров передается в стеке, прямо или косвенно. Считается, что вход в подпрограмму осуществляется с помощью команды CALL, которая помещает адрес возврата в вершину стека и, следовательно, выше параметров.

В тех случаях, когда вставал вопрос о выборе между временем выполнения и использованием памяти, предпочтение отдавалось минимизации времени выполнения. Так, например, было решено дублировать циклы, вместо того, чтобы передавать адреса между стеком и парой регистров.

В дополнение, был выбран подход, минимизирующий число повторяющихся вычислений. Например, в случае индексирования массивов число байтов между начальными адресами элементов, индексы которых отличаются на единицу (известное как *размер* индекса), зависит только от числа байтов, приходящихся на элемент, и от границ массива. Таким образом, размеры различных индексов могут быть вычислены, если известны границы массива. Следовательно, эти размеры используются в качестве параметров для подпрограмм индексирования, так что нет необходимости вычислять их каждый раз при индексировании конкретного массива.

Для большинства коротких подпрограмм было определено время выполнения. Совершенно очевидно, что время выполнения подпрограмм с многими переходами будет зависет от того, по какому пути идет процессор в конкретном случае. Для 8085 это еще более запутано, так как в нем число временных тактов, требуемых для выполнения команд условного перехода, зависит от того, осуществляется переход или нет. Таким образом, часто невозможно определить точное время выполнения. Приведенная здесь документа-

ция всегда содержит по крайней мере один типовой пример, показывающий приблизительное или максимальное время выполнения.

Индикаторы ошибок и специальные случаи должны рассматриваться следующим образом:

1. Подпрограмма должна обеспечивать простую проверку индикатора (такого, как флаг переноса) для определения того, были ли какие-либо ошибки или исключения.

2. Тривиальные случаи, такие как отсутствие элементов в массиве или строки нулевой длины, должны приводить к немедленному выходу из подпрограммы с минимальным-влиянием на исходные данные.

3. Неопределенные данные (такие как максимальная длина строки, равная нулю, или индекс, лежащий вне границы массива) должны приводить к немедленным выходам с минимальным влиянием на исходные данные.

4. Документация должна содержать краткий список ошибок и специальных случаев.

5. В специальных случаях, которые в действительности могут быть удобными для пользователя (такие как задание удаления большего числа символов, чем есть в строке, вместо вычисления точного числа), обработка должна выполняться разумным способом, но все же при этом должна индицироваться ошибка.

Очевидно, что нет метода обработки ошибок и обработки в специальных случаях, который бы полностью удовлетворял, а также был удобным для всех применений. Подход, принятый в данной книге, состоит в том, что лучше дать приемлемый набор подпрограмм, чем не давать его вообще или же считать, что пользователь всегда будет обеспечивать исходные данные в правильной форме.

Приводятся следующие подпрограммы:

ГЛАВА 4

ПРЕОБРАЗОВАНИЕ КОДОВ

4А: ПРЕОБРАЗОВАНИЕ ДВОИЧНЫХ ДАННЫХ В КОД BCD (BN2BCD)

Один байт двоичных данных преобразуется в два байта в коде BCD.

Процедура. Повторяется вычитание 100 из исходных данных для определения цифры разряда сотен, затем повторяется вычитание 10 из остатка для определения цифры разряда десятков и в конце цифра разряда десятков сдвигается влево на четыре позиции и объединяется с цифрой разряда единиц.

Используемые регистры: AF, C, HL.
Время выполнения: максимальное – 491 такт; зависит от количества вычитаний, необходимого для определения цифр разрядов десятков и сотен.
Размер программы: 29 байт.
Память, необходимая для данных: отсутствует.

УСЛОВИЯ НА ВХОДЕ

Двоичные данные в регистре А.

УСЛОВИЯ НА ВЫХОДЕ

Цифра разряда сотен в регистре Н.
Цифры разрядов десятков и единиц в регистре L

ПРИМЕРЫ

1. Данные: (А) = $6E_{16}$ (110 десятичное).
   Результат: (Н) = 01_{16} (цифра разряда сотен),
   (L) = 10_{16} (цифры разрядов десятков и единиц).
2. Данные: (А) = $B7_{16}$ (183 десятичное).
   Результат: (Н) = 01_{16} (цифра разряда сотен),
   (L) = 83_{16} (цифры разрядов десятков и сотен).

```
;                                                              ;
;                                                              ;
;                                                              ;
;                                                              ;
;       ЗАГОЛОВОК:     ПРЕОБРАЗОВАНИЕ ДВОИЧНЫХ ДАННЫХ В КОД BCD ;
;       ИМЯ:           BN2BCD                                  ;
;                                                              ;
;                                                              ;


;                                                              ;
;       НАЗНАЧЕНИЕ:    ПРЕОБРАЗУЕТ ОДИН БАЙТ ДВОИЧНЫХ ДАННЫХ В ДВА ;
;                      БАЙТА ДАННЫХ В КОДЕ BCD                 ;
;                                                              ;
;       ВХОД:          РЕГИСТР A = ДВОИЧНЫЕ ДАННЫЕ             ;
;                                                              ;
;       ВЫХОД:         РЕГИСТР H = СТАРШИЙ БАЙТ ДАННЫХ В КОДЕ BCD ;
;                      РЕГИСТР L = МЛАДШИЙ БАЙТ ДАННЫХ В КОДЕ BCD ;
;
```

```
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ:  AF,C,HL                              ;
;                                                                    ;
;       ВРЕМЯ:            МАКСИМАЛЬНОЕ 491 ТАКТ                      ;
;                                                                    ;
;       РАЗМЕР:           ПРОГРАММА - 29 БАЙТ                        ;
;                                                                    ;
;                                                                    ;

BN2BCD:
        ;ВЫЧИСЛИТЬ ЦИФРУ РАЗРЯДА СОТЕН
        ;РАЗДЕЛИТЬ ДАННЫЕ НА 100
        ; H = ЧАСТНОЕ
        ; A = ОСТАТОК
        MVI     H,OFFH          ;НАЧАЛЬНОЕ ЗНАЧЕНИЕ ЧАСТНОГО РАВНО -1
D100LP:
        INR     H               ;ПРИБАВИТЬ 1 К ЧАСТНОМУ
        SUI     100             ;ВЫЧЕСТЬ 100
        JNC     D100LP          ;ПЕРЕЙТИ, ЕСЛИ А ВСЕ ЕЩЕ БОЛЬШЕ 0
        ADI     100             ;ЕСЛИ НЕТ, ВНОВЬ ПРИБАВИТЬ 100

        ;ВЫЧИСЛИТЬ ЦИФРЫ РАЗРЯДОВ ДЕСЯТКОВ И ЕДИНИЦ
        ;РАЗДЕЛИТЬ НА 10 ОСТАТОК ОТ ЦИФРЫ РАЗРЯДА СОТЕН
        ; L = ЦИФРА РАЗРЯДА ДЕСЯТКОВ
        ; A = ЦИФРА РАЗРЯДА ЕДИНИЦ
        MVI     L,OFFH          ;НАЧАЛЬНОЕ ЗНАЧЕНИЕ ЧАСТНОГО РАВНО -1
D10LP:
        INR     L               ;ПРИБАВИТЬ 1 К ЧАСТНОМУ
        SUI     10              ;ВЫЧЕСТЬ 10
        JNC     D10LP           ;ПЕРЕЙТИ, ЕСЛИ РАЗНОСТЬ ВСЕ ЕЩЕ
                                ; ПОЛОЖИТЕЛЬНАЯ
        ADI     10              ;ЕСЛИ НЕТ, ТО ПРИБАВИТЬ ПОСЛЕДНИЙ
                                ; ДЕСЯТОК

        ;ОБ'ЕДИНИТЬ ЦИФРЫ РАЗРЯДОВ ДЛЯ ЕДИНИЦ И ДЕСЯТКОВ
        MOV     C,A             ;СОХРАНИТЬ ЦИФРУ РАЗРЯДА ЕДИНИЦ В С
        MOV     A,L
        RLC                     ;ПЕРЕСЛАТЬ ДЕСЯТКИ В СТАРШУЮ ПОЛОВИНУ А
        RLC
        RLC
        RLC
        ORA     C               ;ЛОГИЧЕСКОЕ "ИЛИ" С ЦИФРОЙ РАЗРЯДА
                                ; ЕДИНИЦ

        ;ВОЗВРАТ С L = МЛАДШИЙ БАЙТ, H = СТАРШИЙ БАЙТ
        MOV     L,A
        RET


;                                                                    ;
;                                                                    ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                            ;
;                                                                    ;
;                                                                    ;

SC4A:
        ;ПРЕОБРАЗОВАТЬ ОА ШЕСТНАДЦАТЕРИЧНОЕ В 10 В КОДЕ BCD
        MVI     A,OAH
        CALL    BN2BCD          ;H = 0, L = 10H
```

```
;ПРЕОБРАЗОВАТЬ FF ШЕСТНАДЦАТЕРИЧНОЕ В 255 В КОДЕ BCD
MVI     A,0FFH
CALL    BN2BCD            ;H = 02H, L = 55H

;ПРЕОБРАЗОВАТЬ 0 ШЕСТНАДЦАТЕРИЧНЫЙ В 0 В КОДЕ BCD
MVI     A,0
CALL    BN2BCD            ;H = 0, L = 0

JMP     SC4A

END
```

4В. ПРЕОБРАЗОВАНИЕ ДАННЫХ В КОДЕ BCD В ДВОИЧНЫЕ (BCD2BN)

Один байт в коде BCD преобразуется в один байт двоичных данных.

Процедура. С помощью маски выделяется старшая цифра и умножается с использованием сдвигов на 10 (10 = 8 + 2, а умножение на 8 и 2 эквивалентно сдвигам соответственно на три или один разряд влево). Затем произведение прибавляется к младшей цифре.

Используемые регистры: AF, BC.
Время выполнения: 60 тактов.
Размер программы: 14 байт.
Память, необходимая для данных: отсутствует.

УСЛОВИЯ НА ВХОДЕ

Данные в коде BCD в регистре A.

УСЛОВИЯ НА ВЫХОДЕ

Двоичные данные в регистре A.

ПРИМЕРЫ

1. Данные: (A) = 99_{16}.
   Результат: (A) = $63_{16} = 99_{10}$.
2. Данные: (A) = 23_{16}.
   Результат: (A) = $17_{16} = 23_{10}$.

```
;                                                                    ;
;                                                                    ;
;                                                                    ;
;                                                                    ;
;       ЗАГОЛОВОК:        ПРЕОБРАЗОВАНИЕ ДАННЫХ В КОДЕ BCD В ДВОИЧНЫЕ ;
;       ИМЯ:              BCD2BN                                     ;
;                                                                    ;
;                                                                    ;

;                                                                    ;
;       НАЗНАЧЕНИЕ:       ПРЕОБРАЗУЕТ ОДИН БАЙТ ДАННЫХ В КОДЕ BCD В ОДИН ;
;                         БАЙТ ДВОИЧНЫХ ДАННЫХ                       ;
;                                                                    ;
;       ВХОД:             РЕГИСТР A = ДАННЫЕ В КОДЕ BCD              ;
```

```
;                                                                     ;
;       ВЫХОД:              РЕГИСТР A = ДВОИЧНЫЕ ДАННЫЕ               ;
;                                                                     ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ:  AF,BC                                 ;
;                                                                     ;
;       ВРЕМЯ:              60 ТАКТОВ                                 ;
;                                                                     ;
;       РАЗМЕР:             ПРОГРАММА - 14 БАЙТ                       ;
;                                                                     ;
;                                                                     ;

BCD2BN:
        ;УМНОЖИТЬ СТАРШУЮ ПОЛОВИНУ НА 10 И СОХРАНИТЬ ЕЕ
        ; СТАРШАЯ ПОЛОВИНА * 10 = СТАРШАЯ ПОЛОВИНА * ( 8 + 2 )
        MOV     B,A             ;СОХРАНИТЬ НАЧАЛЬНОЕ ЗНАЧЕНИЕ BCD
                                ; В РЕГИСТРЕ B
        ANI     0F0H            ;МАСКИРОВАТЬ СТАРЮУЮ ПОЛОВИНУ
        RRC                     ;СДВИНУТЬ ВПРАВО НА 1 РАЗРЯД
        MOV     C,A             ;C = СТАРШАЯ ПОЛОВИНА * 8
        RRC                     ;СДВИНУТЬ ВПРАВО ЕЩЕ 2 РАЗА
        RRC                     ;A = СТАРШАЯ ПОЛОВИНА * 2
        ADD     C
        MOV     C,A             ;C = СТАРШАЯ ПОЛОВИНА * ( 8 + 2 )

        ;ВЗЯТЬ МЛАДШУЮ ПОЛОВИНУ И ПРИБАВИТЬ ЕЕ
        ; К ДВОИЧНОМУ.ЭКВИВАЛЕНТУ СТАРШЕЙ ПОЛОВИНЫ
        MOV     A,B             ;ВЗЯТЬ ОПЯТЬ НАЧАЛЬНОЕ ЗНАЧЕНИЕ В КОДЕ
                                ; BCD
        ANI     0FH             ;МАСКИРОВАТЬ СТАРШУЮ ПОЛОВИНУ
        ADD     C               ;ДОБАВИТЬ К ДВОИЧНОМУ ЗНАЧЕНИЮ СТАРШУЮ
                                ;  ПОЛОВИНУ
        RET


;                                                                     ;
;                                                                     ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                             ;
;                                                                     ;
;                                                                     ;

SC4B:
        ;ПРЕОБРАЗОВАТЬ 0 В КОДЕ BCD В ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО 0
        MVI     A,0
        CALL    BCD2BN          ;A = 0H

        ;ПРЕОБРАЗОВАТЬ 99 В КОДЕ BCD В ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО 63
        MVI     A,099H

        CALL    BCD2BN          ;A = 63H

        ;ПРЕОБРАЗОВАТЬ 23 В КОДЕ BCD В ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО 17
        MVI     A,23H
        CALL    BCD2BN          ;A = 17H

        JMP     SC4B

        END
```

4C. ПРЕОБРАЗОВАНИЕ ДВОИЧНЫХ ДАННЫХ В ШЕСТНАДЦАТЕРИЧНЫЕ В КОДЕ ASCII (BN2HEX)

Один байт двоичных данных преобразуется в два символа в коде ASCII, соответствующих двум шестнадцатеричным цифрам.

Процедура. С помощью маски выделяется каждая шестнадцатеричная цифра в отдельности и преобразуется в эквивалентный символ ASCII. При этом если цифра десятичная, то используется простое сложение с 30_{16}. Если цифра не десятичная, то необходимо добавить 7, чтобы учесть разницу между символом 9 в коде ASCII (39_{16}) и символом A в коде ASCII (41_{16}).

Используемые регистры: AF, B, HL.
Время выполнения: 162 такта плюс дополнительно семь тактов для каждой недесятичной цифры (8080), или 160 тактов плюс дополнительно четыре такта для каждой недесятичной цифры (8085).
Размер программы: 29 байт.
Память, необходимая для данных: отсутствует.

УСЛОВИЯ НА ВХОДЕ

Двоичные данные в регистре A.

УСЛОВИЯ НА ВЫХОДЕ

Старшая шестнадцатеричная цифра в коде ASCII в регистре H.
Младшая шестнадцатеричная цифра в коде ASCII в регистре L.

ПРИМЕРЫ

1. Данные: (A) = FB_{16}.
   Результат: (H) = 46_{16} (F в коде ASCII),
   (L) = 42_{16} (B в коде ASCII).
2. Данные: (A) = 59_{16}.
   Результат: (H) = 35_{16} (5 в коде ASCII),
   (L) = 39_{16} (9 в коде ASCII).

```
;                                                              ;
;                                                              ;
;                                                              ;
;       ЗАГОЛОВОК:      ПРЕОБРАЗОВАНИЕ ДВОИЧНЫХ ДАННЫХ         ;
;                       В ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО В КОДЕ ASCII ;
;       ИМЯ:            BN2HEX                                 ;
;                                                              ;
;                                                              ;


;                                                              ;
;       НАЗНАЧЕНИЕ:     ПРЕОБРАЗУЕТ ОДИН БАЙТ ДВОИЧНОГО ЧИСЛА В ;
;                       ДВА СИМВОЛА В КОДЕ ASCII               ;
;                                                              ;
;       ВХОД:           РЕГИСТР A = ДВОИЧНОЕ ЧИСЛО             ;
;                                                              ;
;       ВЫХОД:          РЕГИСТР H = СТАРШАЯ ЦИФРА В КОДЕ ASCII ;
;                       РЕГИСТР L = МЛАДШАЯ ЦИФРА В КОДЕ ASCII ;
;                                                              ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ:  AF,B,HL                        ;
```

```
;                                                                ;
;       ВРЕМЯ:          ПРИБЛИЗИТЕЛЬНО 160 ТАКТОВ                ;
;                                                                ;
;       РАЗМЕР:         ПРОГРАММА - 29 БАЙТ                      ;
;                                                                ;
;                                                                ;

BN2HEX:
        ;ПРЕОБРАЗОВАТЬ СТАРШУЮ ПОЛОВИНУ В КОД ASCII
        MOV     B,A             ;СОХРАНИТЬ НАЧАЛЬНОЕ ДВОИЧНОЕ ЗНАЧЕНИЕ
        ANI     0F0H            ;ВЗЯТЬ СТАРШУЮ ПОЛОВИНУ
        RRC                     ;ПЕРЕСЛАТЬ СТАРШУЮ ПОЛОВИНУ В МЛАДШУЮ
        RRC
        RRC
        RRC
        CALL    NASCII          ;ПРЕОБРАЗОВАТЬ СТАРШУЮ ПОЛОВИНУ
                                ; В КОД ASCII
        MOV     H,A             ;ВОЗВРАТИТЬ СТАРШУЮ ПОЛОВИНУ В H

        ;ПРЕОБРАЗОВАТЬ МЛАДШУЮ ПОЛОВИНУ В КОД ASCII
        MOV     A,B
        ANI     0FH             ;ВЗЯТЬ МЛАДШУЮ ПОЛОВИНУ
        CALL    NASCII          ;ПРЕОБРАЗОВАТЬ МЛАДШУЮ ПОЛОВИНУ
                                ; В КОД ASCII
        MOV     L,A             ;ВОЗВРАТИТЬ МЛАДШУЮ ПОЛОВИНУ В L
        RET

        ;--------------------------------
        ;ПОДПРОГРАММА NASCII
        ;НАЗНАЧЕНИЕ:  ПРЕОБРАЗУЕТ ШЕСТНАДЦАТЕРИЧНУЮ ЦИФРУ
        ;             В СИМВОЛ В КОДЕ ASCII
        ;ВХОД:    A = ДВОИЧНОЕ ЧИСЛО В МЛАДШЕЙ ПОЛОВИНЕ БАЙТА
        ;ВЫХОД:   A = СИМВОЛ В КОДЕ ASCII
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: A,F
        ;--------------------------------
NASCII:
        CPI     10
        JC      NAS1            ;ПЕРЕЙТИ, ЕСЛИ СТАРШАЯ ПОЛОВИНА < 10
        ADI     7               ;ИНАЧЕ ПРИБАВИТЬ 7, ЧТОБЫ ПОСЛЕ
                                ; ПРИБАВЛЕНИЯ '0' СИМВОЛЫ БЫЛИ
                                ; В ДИАПАЗОНЕ 'A'..'F'

NAS1:
        ADI     '0'             ;ДОБАВИТЬ '0', ЧТОБЫ ПОЛУЧИТЬ СИМВОЛ
        RET


;                                                                ;
;                                                                ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                        ;
;                                                                ;
;                                                                ;

SC4C:
        ;ПРЕОБРАЗОВАТЬ 0 В '00'
        MVI     A,0
        CALL    BN2HEX          ;H='0'=30H, L='0'=30H
```

```
;ПРЕОБРАЗОВАТЬ ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО FF В 'FF'
MVI      A,0FFH
CALL     BN2HEX              ;H='F'=46H, L='F'=46H

;ПРЕОБРАЗОВАТЬ ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО 23 В '23'
MVI      A,23H
CALL     BN2HEX              ;H='2'=32H, L='3'=33H

JMP      SC4C

END
```

4D. ПРЕОБРАЗОВАНИЕ ШЕСТНАДЦАТЕРИЧНЫХ ДАННЫХ В КОДЕ ASCII В ДВОИЧНЫЕ (HEX2BN)

Два символа ASCII (представляющих две шестнадцатеричные цифры) преобразуются в один байт двоичных данных.

Процедура. Каждый символ ASCII преобразуется в отдельности в шестнадцатеричную цифру. При этом если символ является десятичной цифрой, то используется простое вычитание 30_{16} (0 в коде ASCII). Если же цифра недесятичная, то необходимо вычесть еще 7, чтобы учесть разницу между символом 9 в коде ASCII (39_{16}) и символом A в коде ASCII (41_{16}). Затем старшая цифра сдвигается на 4 разряда влево и объединяется с младшей цифрой. Правильность символов ASCII не проверяется (т. е. является ли в действительности символ ASCII шестнадцатеричной цифрой).

Используемые регистры: AF, B.
Время выполнения: 147 тактов плюс 7 тактов для каждой недесятичной цифры (8080) или 146 тактов плюс 4 такта для каждой недесятичной цифры (8085).
Размер программы: 25 байт.
Память, необходимая для данных: отсутствует.

УСЛОВИЯ НА ВХОДЕ

Старшая цифра в коде ASCII в регистре H, младшая цифра в коде ASCII в регистре L.

УСЛОВИЯ НА ВЫХОДЕ

Двоичные данные в регистре A.

ПРИМЕРЫ

1. Данные: (H) = 44_{16} (D в коде ASCII),
   (L) = 37_{16} (7 в коде ASCII).
   Результат: (A) = $D7_{16}$.
2. Данные: (H) = 31_{16} (1 в коде ASCII),
   (L) = 42_{16} (B в коде ASCII).
   Результат: (A) = $1B_{16}$.

```
;                                                                          ;
;                                                                          ;
;                                                                          ;
;                                                                          ;
;       ЗАГОЛОВОК:      ПРЕОБРАЗОВАНИЕ ШЕСТНАДЦАТЕРИЧНОГО ЧИСЛА В КОДЕ     ;
;                       ASCII В ДВОИЧНОЕ                                   ;
;       ИМЯ:            HEX2BN                                             ;
;                                                                          ;
;                                                                          ;

;                                                                          ;
;       НАЗНАЧЕНИЕ:     ПРЕОБРАЗУЕТ ДВА СИМВОЛА В КОДЕ ASCII В             ;
;                       В ОДИН БАЙТ ДВОИЧНОГО ЧИСЛА                        ;
;                                                                          ;
;       ВХОД:           РЕГИСТР H = СТАРШАЯ ЦИФРА В КОДЕ ASCII             ;
;                       РЕГИСТР L = МЛАДШАЯ ЦИФРА В КОДЕ ASCII             ;
;                                                                          ;
;       ВЫХОД:          РЕГИСТР A = ДВОИЧНОЕ ЧИСЛО                         ;
;                                                                          ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ:  AF,B                                       ;
;                                                                          ;
;       ВРЕМЯ:          ПРИБЛИЗИТЕЛЬНО 147 ТАКТОВ                          ;
;                                                                          ;
;       РАЗМЕР:         ПРОГРАММА - 25 БАЙТ                                ;
;                                                                          ;
;                                                                          ;

HEX2BN:
        MOV     A,L             ;ВЗЯТЬ МЛАДШИЙ СИМВОЛ
        CALL    A2HEX           ;ПРЕОБРАЗОВАТЬ ЕГО В ШЕСТНАДЦАТЕРИЧНУЮ
                                ; ЦИФРУ
        MOV     B,A             ;СОХРАНИТЬ ШЕСТНАДЦАТЕРИЧНОЕ ЗНАЧЕНИЕ
                                ; В РЕГИСТРЕ B
        MOV     A,H             ;ВЗЯТЬ СТАРШИЙ СИМВОЛ
        CALL    A2HEX           ;ПРЕОБРАЗОВАТЬ ЕГО В ШЕСТНАДЦАТЕРИЧНУЮ
                                ; ЦИФРУ
        RLC                     ;СДВИНУТЬ ШЕСТНАДЦАТЕРИЧНОЕ ЗНАЧЕНИЕ
        RLC                     ;  В СТАРШИЕ 4 РАЗРЯДА
        RLC
        RLC
        ORA     B               ;ЛОГИЧЕСКОЕ "ИЛИ" С МЛАДШЕЙ
                                ; ШЕСТНАДЦАТЕРИЧНОЙ ЦИФРОЙ
        RET

        ;-----------------------------------------
        ;ПОДПРОГРАММА: A2HEX
        ;НАЗНАЧЕНИЕ:   ПРЕВРАЩАЕТ ЦИФРУ В КОДЕ ASCII В ШЕСТНАДЦАТЕРИЧНУЮ
        ;ВХОД:  A = ШЕСТНАДЦАТЕРИЧНАЯ ЦИФРА В КОДЕ ASCII
        ;ВЫХОД: A = ДВОИЧНОЕ ЗНАЧЕНИЕ ЦИФРЫ В ASCII
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: A,F
        ;-----------------------------------------
A2HEX:
        SUI     '0'             ;ВЫЧЕСТЬ СМЕЩЕНИЕ СИМВОЛА В КОДЕ ASCII
        CPI     10
        JC      A2HEX1          ;ПЕРЕЙТИ, ЕСЛИ ДЕСЯТИЧНАЯ ЦИФРА
        SUI     7               ;ИНАЧЕ ВЫЧЕСТЬ СМЕЩЕНИЕ ДЛЯ БУКВЫ
A2HEX1:
        RET
```

```
;                                                                  ;
;                                                                  ;
;        ПРИМЕР ВЫПОЛНЕНИЯ                                         ;
;                                                                  ;
;                                                                  ;

SC4D:
        ;ПРЕОБРАЗОВАТЬ 'C7' В ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО C7
        MVI     H,'C'
        MVI     L,'7'
        CALL    HEX2BN          ;A=C7H

        ;ПРЕОБРАЗОВАТЬ '2F' В ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО 2F
        MVI     H,'2'
        MVI     L,'F'
        CALL    HEX2BN          ;A=2FH

        ;ПРЕОБРАЗОВАТЬ '2A' В ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО 2A
        MVI     H,'2'
        MVI     L,'A'
        CALL    HEX2BN          ;A=2AH

        JMP     SC4D

        END
```

4Е. ПРЕОБРАЗОВАНИЕ ДВОИЧНОГО ЧИСЛА В ДЕСЯТИЧНОЕ В КОДЕ ASCII (BN2DEC)

Преобразуется 16-разрядное двоичное число в строку ASCII. Строка содержит длину числа в байтах, знак "минус" в коде ASCII (если необходимо) и цифры в коде ASCII. Заметим, что длина является двоичным числом, а не числом в коде ASCII.

Процедура. Если число отрицательное, то берется его абсолютное значение. Затем абсолютное значение делится на 10 до тех пор, пока частное не станет равным 0. Каждая цифра частного преобразуется в символ ASCII с добавлением 0 в коде ASCII и, если исходное число было отрицательным, с присоединением впереди знака "минус" в коде ASCII.

Используемые регистры: все.
Время выполнения: приблизительно 8400 тактов.
Размер программы: 124 байта.
Память, необходимая для данных: 4 байта в любом месте памяти для указателя буфера (2 байта, начинающиеся по адресу BUFPTR), длины буфера (1 байт по адресу CURLEN) и знака исходного значения числа (1 байт по адресу NGFLAG). Эта память для данных не входит в выходной буфер, длина которого должна составить 7 байт.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес выходного буфера в регистрах H и L.
Преобразуемое значение в регистрах D и E.

УСЛОВИЯ НА ВЫХОДЕ

Порядок данных в буфере:
длина строки в байтах (двоичное число),
знак минус в коде ASCII (если исходное число было отрицательным),
цифра в коде ASCII (первой является старшая цифра числа).

ПРИМЕРЫ

1. Данные: преобразуемое значение = $3EB7_{16}$.
   Результат (в выходном буфере):
   05 (число байтов в буфере),
   31 (1 в коде ASCII),
   36 (6 в коде ASCII),
   30 (0 в коде ASCII),
   35 (5 в коде ASCII),
   35 (5 в коде ASCII).
   Таким образом, $3EB7_{16} = 16055_{10}$.
2. Данные: преобразуемое значение = $FFC8_{16}$.
   Результат (в выходном буфере):
   03 (число байтов в буфере),
   2D (– в коде ASCII),
   35 (5 в коде ASCII),
   36 (6 в коде ASCII).
   Таким образом, $FFC8_{16} = -56_{10}$, если рассматривается как дополнение до двух числа со знаком.

```
;                                                               ;
;                                                               ;
;                                                               ;
;    ЗАГОЛОВОК:     ПРЕОБРАЗОВАНИЕ ДВОИЧНОГО ЧИСЛА В ДЕСЯТИЧНОЕ ;
;                   В КОДЕ ASCII                                ;
;    ИМЯ:           BN2DEC                                      ;
;                                                               ;
;                                                               ;


;                                                               ;
;    НАЗНАЧЕНИЕ:    ПРЕОБРАЗУЕТ 16-РАЗРЯДНОЕ ДВОИЧНОЕ ЧИСЛО     ;
;                   СО ЗНАКОМ В ДАННЫЕ В КОДЕ ASCII             ;
;                                                               ;
;    ВХОД:          РЕГИСТР H = СТАРШИЙ БАЙТ АДРЕСА ВЫХОДНОГО   ;
;                               БУФЕРА                          ;
;                   РЕГИСТР L = МЛАДШИЙ БАЙТ АДРЕСА ВЫХОДНОГО   ;
;                               БУФЕРА                          ;
;                   РЕГИСТР D = СТАРШИЙ БАЙТ ПРЕОБРАЗУЕМОГО     ;
;                               ЗНАЧЕНИЯ                        ;
;                   РЕГИСТР E = МЛАДШИЙ БАЙТ ПРЕОБРАЗУЕМОГО     ;
;                               ЗНАЧЕНИЯ                        ;
;                                                               ;
;    ВЫХОД:         ПЕРВЫЙ БАЙТ БУФЕРА СОДЕРЖИТ ДЛИНУ,          ;
;                   ЗА НЕЙ СЛЕДУЮТ СИМВОЛЫ                      ;
;                                                               ;
;    ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ:  ВСЕ                                ;
;                                                               ;
;    ВРЕМЯ:         ПРИБЛИЗИТЕЛЬНО 8400 ТАКТОВ                  ;
```

```
;
;       РАЗМЕР:          ПРОГРАММА - 124 БАЙТА                              ;
;                        ДАННЫЕ    -   4 БАЙТА                              ;
;                                                                           ;
;                                                                           ;

BN2DEC:
        ;СОХРАНИТЬ ПАРАМЕТРЫ
        SHLD    BUFPTR          ;ЗАПОМНИТЬ УКАЗАТЕЛЬ БУФЕРА
        XCHG                    ;HL = ПРЕОБРАЗУЕМОЕ ЗНАЧЕНИЕ
        MVI     A,0
        STA     CURLEN          ;ТЕКУЩАЯ ДЛИНА БУФЕРА РАВНА 0
        MOV     A,H
        STA     NGFLAG          ;СОХРАНИТЬ ЗНАК ЗНАЧЕНИЯ
        ORA     A               ;УСТАНОВИТЬ ФЛАГИ
        JP      CNVERT          ;ПЕРЕЙТИ, ЕСЛИ ЗНАЧЕНИЕ ПОЛОЖИТЕЛЬНОЕ
        SUB     A               ;ИНАЧЕ ВЗЯТЬ АБСОЛЮТНОЕ ЗНАЧЕНИЕ
        SUB     L               ; (0 - ЗНАЧЕНИЕ)
        MOV     L,A
        SBB     A               ;ПРОДВИНУТЬ ЗАЕМ
        SUB     H
        MOV     H,A

        ;ПРЕОБРАЗОВАТЬ ЗНАЧЕНИЕ В СТРОКУ
CNVERT:
        ;HL := HL DIV 10  (ДЕЛИМОЕ,ЧАСТНОЕ)
        ;DE := HL MOD 10  (ОСТАТОК)
        MVI     E,0             ;ОСТАТОК = 0
        MVI     B,16            ;16 РАЗРЯДОВ В ДЕЛИМОЕ
        ORA     A               ;ДЛЯ НАЧАЛА ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА
DVLOOP:
        ;СДВИНУТЬ СЛЕДУЮЩИЙ РАЗРЯД ЧАСТНОГО В РАЗРЯД 0 ДЕЛИМОГО
        ;СДВИНУТЬ СЛЕДУЮЩИЙ СТАРШИЙ ПО ЗНАЧЕНИЮ РАЗРЯД ДЕЛИМОГО В
        ; НАИМЕНЕЕ ЗНАЧИМЫЙ РАЗРЯД ОСТАТКА
        ;HL СОДЕРЖИТ КАК ДЕЛИМОЕ, ТАК И ЧАСТНОЕ. ЧАСТНОЕ СДВИГАЕТСЯ
        ; ВНУТРЬ НА СТОЛЬКО ЖЕ, НА СКОЛЬКО ДЕЛИМОЕ СДВИГАЕТСЯ НАРУЖУ
        ;E - ОСТАТОК

        ;ВЫПОЛНИТЬ 24-РАЗРЯДНЫЙ СДВИГ ВЛЕВО, СДВИГАЯ
        ; ФЛАГ ПЕРЕНОСА В L, L В H, H В E
        MOV     A,L
        RAL                     ;ФЛАГ ПЕРЕНОСА (СЛЕДУЮЩИЙ РАЗРЯД ЧАСТНОГО)
        MOV     L,A             ; В РАЗРЯД 0, РАЗРЯД 7 - ВО ФЛАГ ПЕРЕНОСА
        MOV     A,H
        RAL                     ;СДВИНУТЬ ДРУГОЙ БАЙТ ДЕЛИМОГО
        MOV     H,A             ; И ЧАСТНОГО
        MOV     A,E
        RAL                     ;СДВИНУТЬ СЛЕДУЮЩИЙ РАЗРЯД ДЕЛИМОГО
        MOV     E,A             ; В ОСТАТОК

        ;ЕСЛИ ОСТАТОК РАВЕН ИЛИ БОЛЬШЕ 10, ТО СЛЕДУЮЩИЙ РАЗРЯД
        ; ЧАСТНОГО РАВЕН 1 (ЭТОТ РАЗРЯД ПОМЕЩАЕТСЯ ВО ФЛАГ ПЕРЕНОСА)
        SUI     10              ;ВЫЧЕСТЬ 10 ИЗ ОСТАТКА
        CMC                     ;ИНВЕРТИРОВАТЬ ФЛАГ ПЕРЕНОСА
                                ; (ЭТО СЛЕДУЮЩИЙ РАЗРЯД ЧАСТНОГО)
        JNC     DECCNT          ;ПЕРЕЙТИ, ЕСЛИ ОСТАТОК МЕНЬШЕ 10
        MOV     E,A             ;ИНАЧЕ СДЕЛАТЬ ОСТАТОК = РАЗНИЦА
                                ; МЕЖДУ ПРЕДЫДУЩИМ ОСТАТКОМ И 10
```

```
DECCNT:
        DCR     B
        JNZ     DVLOOP          ;ПРОДОЛЖАТЬ, ПОКА НЕ БУДУТ ОБРАБОТАНЫ
                                ; ВСЕ РАЗРЯДЫ

        ;СДВИНУТЬ РЕЗУЛЬТАТ ПОСЛЕДНЕГО ПЕРЕНОСА В ЧАСТНОЕ
        RAL                     ;ПОСЛЕДНИЙ РАЗРЯД ЧАСТНОГО В РАЗРЯД 0
        ANI     1               ;УДАЛИТЬ ВСЕ ОСТАЛЬНЫЕ РАЗРЯДЫ
        DAD     H               ;СДВИНУТЬ ЧАСТНОЕ ВЛЕВО
        ORA     L               ;ВЫПОЛНИТЬ ЛОГИЧЕСКУЮ ОПЕРАЦИЮ "ИЛИ"
                                ; С РАЗРЯДОМ 0
        MOV     L,A             ;ПЕРЕСЛАТЬ НАЗАД В L

        ;ВСТАВИТЬ СЛЕДУЮЩИЙ СИМВОЛ
CHINS:
        MOV     A,E
        ADI     '0'             ;ПРЕОБРАЗОВАТЬ 0..9 В '0'..'9' В КОДЕ
        CALL    INSERT          ; ASCII

        ;ЕСЛИ ЧАСТНОЕ НЕ РАВНО 0, ПРОДОЛЖАТЬ ДЕЛЕНИЕ
        MOV     A,H             ;ПРОВЕРИТЬ ЧАСТНОЕ
        ORA     L
        JNZ     CNVERT

EXIT:
        LDA     NGFLAG
        ORA     A
        JP      POS             ;ПЕРЕЙТИ, ЕСЛИ ИСХОДНОЕ ЗНАЧЕНИЕ
                                ; ПОЛОЖИТЕЛЬНОЕ
        MVI     A,'-'           ;ИНАЧЕ
        CALL    INSERT          ; ВСТАВИТЬ В НАЧАЛЕ ЗНАК МИНУС

POS:
        RET

        ;--------------------------------
        ;ПОДПРОГРАММА: INSERT
        ;НАЗНАЧЕНИЕ: ВСТАВКА СИМВОЛА ИЗ РЕГИСТРА A ПЕРЕД БУФЕРОМ
        ;ВХОД:  CURLEN = ДЛИНА БУФЕРА
        ;       BUFPTR = ТЕКУЩИЙ АДРЕС ПОСЛЕДНЕГО СИМВОЛА В БУФЕРЕ
        ;ВЫХОД: РЕГИСТР A ВСТАВЛЯЕТСЯ НЕПОСРЕДСТВЕННО ПОСЛЕ БАЙТА ДЛИНЫ
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF,B,C,D,E
        ;--------------------------------
INSERT:
        PUSH    H               ;СОХРАНИТЬ HL
        MOV     C,A             ;СОХРАНИТЬ СИМВОЛ В C

        ;ПЕРЕСЛАТЬ БУФЕР ВПРАВО НА ОДИН СИМВОЛ
        LHLD    BUFPTR          ;ВЗЯТЬ УКАЗАТЕЛЬ БУФЕРА
        MOV     D,H             ;DE = АДРЕС ИСТОЧНИКА (ТЕКУЩИЙ КОНЕЦ БУФЕРА)
        MOV     E,L
        INX     H               ;HL = АДРЕС НАЗНАЧЕНИЯ (ТЕКУЩИЙ КОНЕЦ + 1)
        SHLD    BUFPTR          ;ЗАПОМНИТЬ НОВЫЙ УКАЗАТЕЛЬ БУФЕРА
        LDA     CURLEN
        ORA     A               ;ПРОВЕРИТЬ CURLEN НА = 0
        JZ      EXITMR          ;ПЕРЕЙТИ, ЕСЛИ 0 (НИЧЕГО НЕ ПЕРЕСЫЛАЯ)
                                ; ПРОСТО ЗАПОМНИТЬ СИМВОЛ
        MOV     B,A             ;B = СЧЕТЧИК ЦИКЛА
```

```
MVELP:
        XCHG                            ;HL = АДРЕС ИСТОЧНИКА
        MOV     A,M                     ;ВЗЯТЬ СЛЕДУЮЩИЙ СИМВОЛ
        XCHG
        MOV     M,A                     ;ЗАПОМНИТЬ ЕГО
        DCX     H                       ;УМЕНЬШИТЬ АДРЕС НАЗНАЧЕНИЯ
        DCX     D                       ;УМЕНЬШИТЬ АДРЕС ИСТОЧНИКА
        DCR     B                       ;УМЕНЬШИТЬ СЧЕТЧИК
        JNZ     MVELP                   ;ПРОДОЛЖАТЬ, ПОКА НЕ БУДУТ ПЕРЕСЛАНЫ
                                        ;  ВСЕ БАЙТЫ

EXITMR:
        MOV     A,C                     ;ВЗЯТЬ ВСТАВЛЯЕМЫЙ СИМВОЛ
        MOV     M,A                     ;ВСТАВИТЬ СИМВОЛ ПЕРЕД БУФЕРОМ
        LDA     CURLEN                  ;УВЕЛИЧИТЬ ТЕКУЩУЮ ДЛИНУ НА 1
        INR     A
        STA     CURLEN
        DCX     H                       ;УКАЗАТЬ НА БАЙТ ДЛИНЫ В БУФЕРЕ
        MOV     M,A                     ;ОБНОВИТЬ ЕГО
        POP     H                       ;ВОССТАНОВИТЬ HL
        RET

        ;ДАННЫЕ
BUFPTR: DS      2                       ;АДРЕС ПОСЛЕДНЕГО СИМВОЛА В БУФЕРЕ
CURLEN: DS      1                       ;ТЕКУЩАЯ ДЛИНА БУФЕРА
NGFLAG: DS      1                       ;ЗНАК ИСХОДНОГО ЗНАЧЕНИЯ


;
;
;       ПРИМЕР ВЫПОЛНЕНИЯ
;
;

SC4E:
        ;ПРЕОБРАЗОВАТЬ 0 В '0'
        LXI     H,BUFFER                ;HL = БАЗОВЫЙ АДРЕС БУФЕРА
        LXI     D,0                     ;DE = 0
        CALL    BN2DEC                  ;ПРЕОБРАЗОВАННЫЙ
                                        ; БУФЕР ДОЛЖЕН БЫТЬ = '0'

        ;ПРЕОБРАЗОВАТЬ 32767 В '32767'
        LXI     H,BUFFER                ;HL = БАЗОВЫЙ АДРЕС БУФЕРА
        LXI     D,32767                 ;DE = 32767
        CALL    BN2DEC                  ;ПРЕОБРАЗОВАННЫЙ
                                        ; БУФЕР ДОЛЖЕН БЫТЬ = '32767'

        ;ПРЕОБРАЗОВАТЬ -32768 В '-32768'
        LXI     H,BUFFER                ;HL = БАЗОВЫЙ АДРЕС БУФЕРА
        LXI     D,-32768                ;DE = -32768
        CALL    BN2DEC                  ;ПРЕОБРАЗОВАННЫЙ
                                        ; БУФЕР ДОЛЖЕН БЫТЬ = '-32768'

        JMP     SC4E

BUFFER: DS      7                       ;БУФЕР ДЛИНОЙ 7 БАЙТ

        END
```

4F. ПРЕОБРАЗОВАНИЕ ДЕСЯТИЧНОГО ЧИСЛА В КОДЕ ASCII В ДВОИЧНОЕ (DEC2BN)

Преобразуются строка в коде ASCII, содержащая длину числа (в байтах), знак + или –, если он есть, и ряд цифр в коде ASCII, в два байта двоичного числа. Заметим, что длина является обычным двоичным числом, а не числом в коде ASCII.

Процедура. Если первым символом в коде ASCII является знак "минус", то устанавливается флаг, если же плюс – этот символ пропускается. Затем каждая последующая цифра преобразуется в десятичное число, при этом вычитается 0 в коде ASCII, умножаются предыдущие цифры на 10 (с учетом того, что 10 = 8 + 2, так что умножение на 10 может быть сведено к сдвигам влево и сложениям) и добавляется новая цифра к полученному произведению. Если исходное число было отрицательным, то в конце результат вычитается из 0. Если в начале строки обнаруживается не знак числа или цифра или же в строке находится не цифра, то осуществляется немедленный выход из программы с установленным знаком переноса.

Используемые регистры: все.

Время выполнения: приблизительно 160 тактов на байт плюс максимум 201 такт.

Размер программы: 88 байт.

Память, необходимая для данных: 1 байт где-либо в ОЗУ (адрес NGFLAG) для флага, указывающего на знак числа.

Специальные случаи:

1. Если обнаруживается не знак числа в начале или не десятичная цифра в начале или в строке, то осуществляется немедленный выход с флагом переноса, установленным в 1. В этом случае результат в регистрах HL неправильный.

2. Если строка содержит только знак числа (+ или – в коде ASCII), то производится немедленный выход с флагом переноса, установленным в 1, и результатом, равным 0.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес строки в регистрах H и L.

УСЛОВИЯ НА ВЫХОДЕ

Двоичное значение в регистрах H и L.
Флаг переноса равен 0, если строка правильная; флаг переноса равен 1, если строка содержит неправильный символ.
Заметим, что результат равен 16-разрядному числу со знаком, являющемуся дополнением до двух.

ПРИМЕРЫ

1. Данные: строка содержит:
   04 (число байтов в строке),
   31 (1 в коде ASCII),
   32 (2 в коде ASCII),
   33 (3 в коде ASCII),
   34 (4 в коде ASCII).

   Таким образом: число равно $+1234_{10}$.

Результат: (H) = 04_{16} (старший байт двоичного числа),
(L) = $D2_{16}$ (младший байт двоичного числа).

Таким образом, число $+1234_{10} = 04D2_{16}$.

2. Данные: строка содержит:
06 (число байт в строке),
2D (– в коде ASCII),
33 (3 в коде ASCII),
32 (2 в коде ASCII),
37 (7 в коде ASCII),
35 (5 в коде ASCII),
30 (0 в коде ASCII).

Таким образом, число равно -32750_{10}.

Результат: (H) = 80_{16} (старший байт двоичного числа),
(L) = 12_{16} (младший байт двоичного числа).

Таким образом, число $-32750_{10} = 8012_{16}$.

```
;                                                                        ;
;                                                                        ;
;                                                                        ;
;                                                                        ;
;       ЗАГОЛОВОК:        ПРЕОБРАЗОВАНИЕ ДЕСЯТИЧНОГО ЧИСЛА В КОДЕ ASCII   ;
;                         В ДВОИЧНОЕ                                      ;
;       ИМЯ:              DEC2BN                                          ;
;                                                                        ;
;                                                                        ;


;                                                                        ;
;       НАЗНАЧЕНИЕ:       ПРЕОБРАЗУЕТ СИМВОЛЫ В КОДЕ ASCII                ;
;                         В ДВА БАЙТА ДВОИЧНЫХ ДАННЫХ                     ;
;                                                                        ;
;       ВХОД:             РЕГИСТР H = СТАРШИЙ БАЙТ АДРЕСА ВХОДНОГО БУФЕРА ;
;                         РЕГИСТР L = МЛАДШИЙ БАЙТ АДРЕСА ВХОДНОГО БУФЕРА ;
;                                                                        ;
;       ВЫХОД:            РЕГИСТР H = СТАРШИЙ БАЙТ ЗНАЧЕНИЯ               ;
;                         РЕГИСТР L = МЛАДШИЙ БАЙТ ЗНАЧЕНИЯ               ;
;                         ЕСЛИ НЕТ ОШИБОК, ТО                             ;
;                          ФЛАГ ПЕРЕНОСА = 0                              ;
;                         ИНАЧЕ                                           ;
;                          ФЛАГ ПЕРЕНОСА = 1                              ;
;                                                                        ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ:  ВСЕ                                       ;
;                                                                        ;
;       ВРЕМЯ:            ПРИБЛИЗИТЕЛЬНО 160 ТАКТОВ НА БАЙТ ПЛЮС          ;
;                         МАКСИМУМ 201 ТАКТ                               ;
;                                                                        ;
;       РАЗМЕР:           ПРОГРАММА - 88 БАЙТ                             ;
;                         ДАННЫЕ    -  1 БАЙТ                             ;
;                                                                        ;
;                                                                        ;
DEC2BN:
        ;ИНИЦИАЛИЗАЦИЯ - СОХРАНИТЬ ДЛИНУ, ОЧИСТИТЬ ЗНАК И ЗНАЧЕНИЕ
        MOV     A,M             ;СОХРАНИТЬ ДЛИНУ В РЕГИСТРЕ B
        MOV     B,A
        INX     H               ;УКАЗАТЬ НА БАЙТ ПОСЛЕ ДЛИНЫ
        SUB     A
```

```
        STA     NGFLAG              ;СЧИТАТЬ ЧИСЛО ПОЛОЖИТЕЛЬНЫМ
        LXI     D,0                 ;НАЧАТЬ СО ЗНАЧЕНИЯ = 0

        ;ПРОВЕРИТЬ, ПУСТ ЛИ БУФЕР
        ORA     B                   ;ДЛИНА БУФЕРА НУЛЬ?
        JZ      EREXIT              ;ДА, ВЫЙТИ СО ЗНАЧЕНИЕМ = 0

        ;ПРОВЕРИТЬ НА ЗНАК МИНУС ИЛИ ПЛЮС В НАЧАЛЕ ЧИСЛА
INIT1:
        MOV     A,M                 ;ВЗЯТЬ ПЕРВЫЙ СИМВОЛ
        CPI     '-'                 ;ЭТО ЗНАК МИНУС?
        JNZ     PLUS                ;НЕТ, ПЕРЕЙТИ
        MVI     A,0FFH              ;ДА, СДЕЛАТЬ ЗНАК ОТРИЦАТЕЛЬНЫМ
        STA     NGFLAG
        JMP     SKIP                ;ПРОПУСТИТЬ ЗНАК МИНУС

PLUS:
        CPI     '+'                 ;ПЕРВЫЙ СИМВОЛ ЗНАК ПЛЮС?;
        JNZ     CHKDIG              ;ДА, НАЧАТЬ ПРЕОБРАЗОВАНИЕ
SKIP:   INX     H                   ;ПРОПУСТИТЬ БАЙТ СО ЗНАКОМ
        DCR     B                   ;УМЕНЬШИТЬ СЧЕТЧИК
        JZ      EREXIT              ;ВЫХОД ПО ОШИБКЕ, ЕСЛИ В БУФЕРЕ
                                    ; ТОЛЬКО ЗНАК

        ;ЦИКЛ ПРЕОБРАЗОВАНИЯ
        ; ПРОДОЛЖАЕТСЯ ДО ТЕХ ПОР, ПОКА БУФЕР НЕ БУДЕТ ПУСТЫМ,
        ; ИЛИ НЕ БУДЕТ НАЙДЕН НЕЦИФРОВОЙ СИМВОЛ
CNVERT:
        MOV     A,M                 ;ВЗЯТЬ СЛЕДУЮЩИЙ СИМВОЛ
CHKDIG: SUI     '0'
        JC      EREXIT              ;ОШИБКА, ЕСЛИ < '0' (НЕ ЦИФРА)
        CPI     9+1
        JNC     EREXIT              ;ОШИБКА, ЕСЛИ > '9' (НЕ ЦИФРА)
        MOV     C,A                 ;СИМВОЛ - ЦИФРА, СОХРАНИТЬ ЕГО

        ;ПРАВИЛЬНАЯ ДЕСЯТИЧНАЯ ЦИФРА, ПОЭТОМУ
        ;   ЗНАЧЕНИЕ := ЗНАЧЕНИЕ * 10
        ; = ЗНАЧЕНИЕ * (8 + 2)
        ; = (ЗНАЧЕНИЕ * 8) + (ЗНАЧЕНИЕ * 2)
        PUSH    H                   ;СОХРАНИТЬ УКАЗАТЕЛЬ БУФЕРА
        XCHG                        ;HL = ЗНАЧЕНИЕ
        DAD     H                   ; * 2
        MOV     E,L                 ;СОХРАНИТЬ УДВОЕННОЕ ЗНАЧЕНИЕ В DE
        MOV     D,H
        DAD     H                   ; * 4
        DAD     H                   ; * 8
        DAD     D                   ;ЗНАЧЕНИЕ = ЗНАЧЕНИЕ * 10

        ;ДОБАВИТЬ СЛЕДУЮЩУЮ ЦИФРУ
        ; ЗНАЧЕНИЕ := ЗНАЧЕНИЕ + ЦИФРА
        MOV     E,C                 ;ПЕРЕСЛАТЬ СЛЕДУЮЩУЮ ЦИФРУ В E
        MVI     D,0                 ; СТАРШИЙ БАЙТ РАВЕН 0
        DAD     D                   ;ДОБАВИТЬ ЦИФРУ К ЗНАЧЕНИЮ
        XCHG                        ;DE = ЗНАЧЕНИЕ
        POP     H                   ;УКАЗАТЬ НА СЛЕДУЮЩИЙ СИМВОЛ
        INX     H
        DCR     B
        JNZ     CNVERT              ;ПРОДОЛЖИТЬ ПРЕОБРАЗОВАНИЕ
```

```
        ;ПРЕОБРАЗОВАНИЕ ЗАКОНЧЕНО, ПРОВЕРИТЬ ЗНАК
        XCHG                        ;HL = ЗНАЧЕНИЕ
        LDA     NGFLAG
        ORA     A
        JP      OKEXIT              ;ПЕРЕЙТИ, ЕСЛИ ЗНАЧЕНИЕ
                                    ; БЫЛО ПОЛОЖИТЕЛЬНЫМ
        SUB     A                   ;ИНАЧЕ ЗАМЕНИТЬ РЕЗУЛЬТАТ
        SUB     L                   ; НА РЕЗУЛЬТАТ СО ЗНАКОМ МИНУС
        MOV     L,A
        SBB     A                   ;ПРОДВИНУТЬ ЗАЕМ
        SUB     H
        MOV     H,A

        ;НЕТ ОШИБОК, ВЫЙТИ С ОЧИЩЕННЫМ ФЛАГОМ ПЕРЕНОСА
OKEXIT:
        ORA     A                   ;ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА
        RET
        ;ОШИБКА, ВЫЙТИ С УСТАНОВЛЕННЫМ ФЛАГОМ ПЕРЕНОСА
EREXIT:
        XCHG                        ;HL = ЗНАЧЕНИЕ
        STC                         ;УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА
        RET

        ;ДАННЫЕ
NGFLAG: DS      1                   ;ЗНАК ЧИСЛА

;                                                                    ;
;                                                                    ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                            ;
;                                                                    ;
;                                                                    ;

SC4F:
        ;ПРЕОБРАЗОВАТЬ '1234' В ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО 04D2
        LXI     H,S1                ;HL = БАЗОВЫЙ АДРЕС S1
        CALL    DEC2BN
                                    ;H = 04, L = ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО D2

        ;ПРЕОБРАЗОВАТЬ '+32767' В ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО 7FFF
        LXI     H,S2                ;HL = БАЗОВЫЙ АДРЕС S2
        CALL    DEC2BN
                                    ;H = ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО 7F
                                    ;L = ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО FF

        ;ПРЕОБРАЗОВАТЬ '-32768' В ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО 8000
        LXI     H,S3                ;HL = БАЗОВЫЙ АДРЕС S3
        CALL    DEC2BN
                                    ;H = ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО 80
                                    ;L = ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО 00

        JMP     SC4F

S1:     DB      4,'1234'
S2:     DB      6,'+32767'
S3:     DB      6,'-32768'

        END
```

4G. ТРАНСЛЯЦИЯ СТРОЧНЫХ БУКВ В ПРОПИСНЫЕ

Строчная буква в коде ASCII преобразуется в эквивалентную прописную букву.

Процедура. С помощью сравнения определяется, являются ли данные строчной буквой в коде ASCII. Если да, то из данных вычитается 20_{16}, и они преобразуются, таким образом, в строчный эквивалент. Если нет, то данные остаются без изменения.

Используемые регистры: AF.
Время выполнения: 51 такт, если исходный символ является строчной буквой; в противном случае меньше.
Размер программы: 13 байт.
Память, необходимая для данных: отсутствует.

УСЛОВИЯ НА ВХОДЕ

Символ в регистре A.

УСЛОВИЯ НА ВЫХОДЕ

Если в регистре A находится строчная буква в коде ASCII, то в A возвращается эквивалентная прописная буква. Во всех других случаях регистр A остается без изменения.

ПРИМЕРЫ

1. Данные: (A) = 62_{16} (b в коде ASCII).
   Результат: (A) = 42_{16} (B в коде ASCII).
2. Данные: (A) = 54_{16} (T в коде ASCII).
   Результат: (A) = 54_{16} (T в коде ASCII).

```
;                                                                    ;
;                                                                    ;
;                                                                    ;
;                                                                    ;
;       ЗАГОЛОВОК:      ТРАНСЛЯЦИЯ СТРОЧНЫХ БУКВ В ПРОПИСНЫЕ         ;
;       ИМЯ:            LC2UC                                        ;
;                                                                    ;
;                                                                    ;
;                                                                    ;
;       НАЗНАЧЕНИЕ:     ПРЕОБРАЗУЕТ, ЕСЛИ НЕОБХОДИМО, ОДНУ БУКВУ В   ;
;                       КОДЕ ASCII ИЗ СТРОЧНОЙ В ПРОПИСНУЮ           ;
;                                                                    ;
;       ВХОД:           РЕГИСТР A = СТРОЧНАЯ БУКВА В КОДЕ ASCII      ;
;                                                                    ;
;       ВЫХОД:          РЕГИСТР A = ПРОПИСНАЯ БУКВА В КОДЕ ASCII, ЕСЛИ ;
;                                   БЫЛА СТРОЧНОЙ, ИНАЧЕ A НЕ ИЗМЕНЯЕТСЯ;
;                                                                    ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ:   AF                                  ;
;                                                                    ;
;       ВРЕМЯ:          51 ТАКТ, ЕСЛИ В A СТРОЧНАЯ БУКВА, ИНАЧЕ МЕНЬШЕ ;
;                                                                    ;
;       РАЗМЕР:         ПРОГРАММА - 13 БАЙТ                          ;
;                       ДАННЫЕ    - ОТСУТСТВУЮТ                      ;
;                                                                    ;
;                                                                    ;

LC2UC:
        CPI     'a'
        JC      EXIT                ;ПЕРЕЙТИ, ЕСЛИ < 'a' (НЕ СТРОЧНАЯ)
```

```
        CPI     'z'+1
        JNC     EXIT              ;ПЕРЕЙТИ, ЕСЛИ > 'z' (НЕ СТРОЧНАЯ)
        SUI     'a'-'A'           ;ЗАМЕНИТЬ 'a'..'z' НА 'A'..'Z'
EXIT:
        RET



;                                                                  ;
;                                                                  ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                          ;
;                                                                  ;
;                                                                  ;

SC4G:
        ;ПРЕОБРАЗОВАТЬ СТРОЧНУЮ БУКВУ E В ПРОПИСНУЮ
        MVI     A,'e'
        CALL    LC2UC             ;A='E'=45H

        ;ПРЕОБРАЗОВАТЬ СТРОЧНУЮ БУКВУ Z В ПРОПИСНУЮ
        MVI     A,'z'
        CALL    LC2UC             ;A='Z'=5AH

        ;ПРЕОБРАЗОВАТЬ ПРОПИСНУЮ БУКВУ A В ПРОПИСНУЮ
        MVI     A,'A'
        CALL    LC2UC             ;A='A'=41H
        JMP     SC4G

        END
```

4Н. ПРЕОБРАЗОВАНИЕ СИМВОЛА В КОДЕ ASCII В ЕГО ЭКВИВАЛЕНТ В КОДЕ EBCDIC (ASC2EB)

Символ в коде ASCII преобразуется в его эквивалент в коде EBCDIC.

Процедура. Используется простой поиск в таблице, при этом данные являются индексом, а адрес EBCDIC – базой. Символы ASCII, которые печатаются, но не имеют эквивалента EBCDIC, транслируются в пробел в конце EBCDIC (40_{16}); символы ASCII, которые не печатаются и не имеют эквивалента EBCDIC, транслируются в EBCDIC NUL (00_{16}).

Используемые регистры:
Время выполнения: 55 тактов.
Размер программы: 11 байт плюс 128 байт для таблицы преобразования.
Память, необходимая для данных: отсутствует.

УСЛОВИЯ НА ВХОДЕ

Символ в коде ASCII в регистре A.

УСЛОВИЯ НА ВЫХОДЕ

Эквивалент в коде EBCDIC в регистре A.

ПРИМЕРЫ

1. Данные: (A) = 35_{16} (5 в коде ASCII).
   Результат: (A) = $F5_{16}$ (5 в коде EBCDIC).
2. Данные: (A) = 77_{16} (w в коде ASCII).
   Результат: (A) = $A6_{16}$ (w в коде EBCDIC).
3. Данные: (A) = $2A_{16}$ (* в коде ASCII).
   Результат: (A) = $5C_{16}$ (* в коде EBCDIC).

```
;                                                                              ;
;                                                                              ;
;                                                                              ;
;                                                                              ;
;       ЗАГОЛОВОК:      ПРЕОБРАЗОВАНИЕ СИМВОЛА В КОДЕ ASCII В ЕГО              ;
;                       ЭКВИВАЛЕНТ В КОДЕ EBCDIC                               ;
;       ИМЯ:            ASC2EB                                                 ;
;                                                                              ;
;                                                                              ;
;                                                                              ;
;       НАЗНАЧЕНИЕ:     ПРЕОБРАЗУЕТ СИМВОЛ В КОДЕ ASCII В ЕГО                  ;
;                       ЭКВИВАЛЕНТ В КОДЕ EBCDIC                               ;
;                                                                              ;
;       ВХОД:           РЕГИСТР A = СИМВОЛ В КОДЕ ASCII                        ;
;                                                                              ;
;       ВЫХОД:          РЕГИСТР A = СИМВОЛ В КОДЕ EBCDIC                       ;
;                                                                              ;
;                                                                              ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ:   AF,DE,HL                                      ;
;                                                                              ;
;       ВРЕМЯ:          55 ТАКТОВ                                              ;
;                                                                              ;
;       РАЗМЕР:         ПРОГРАММА - 11 БАЙТ                                    ;
;                       ДАННЫЕ    - 128 БАЙТ ДЛЯ ТАБЛИЦЫ                       ;
;                                                                              ;
;                                                                              ;

ASC2EB:
        LXI     H,EBCDIC        ;ВЗЯТЬ БАЗОВЫЙ АДРЕС ТАБЛИЦЫ EBCDIC
        ANI     01111111B       ;УСТАНОВИТЬ РАЗРЯД 7 = 0
        MOV     E,A             ;ИСПОЛЬЗОВАТЬ КОД ASCII КАК ИНДЕКС
                                ; В ТАБЛИЦЕ EBCDIC
        MVI     D,0
        DAD     D
        MOV     A,M             ;ВЗЯТЬ СИМВОЛ В КОДЕ EBCDIC
        RET

;ТАБЛИЦА СООТВЕТСТВИЯ СИМВОЛОВ В КОДЕ ASCII И EBCDIC
; СИМВОЛЫ ASCII, КОТОРЫЕ ПЕЧАТАЮТСЯ, НО НЕ ИМЕЮТ ЭКВИВАЛЕНТА В КОДЕ
; EBCDIC, ТРАНСЛИРУЮТСЯ В ПРОБЕЛ В КОДЕ EBCDIC (040H), А СИМВОЛЫ ASCII,
; КОТОРЫЕ НЕ ПЕЧАТАЮТСЯ И НЕ ИМЕЮТ ЭКВИВАЛЕНТА В КОДЕ EBCDIC,
; ТРАНСЛИРУЮТСЯ В NUL В КОДЕ EBCDIC (000H)
EBCDIC:
        ;       NUL  SOH  STX  ETX  EOT  ENQ  ACK  BEL          ;ASCII
        DB      000H,001H,002H,003H,037H,02DH,02EH,02FH         ;EBCDIC
        ;       BS   HT   LF   VT   FF   CR   SO   SI           ;ASCII
        DB      016H,005H,025H,00BH,00CH,00DH,00EH,00FH         ;EBCDIC
        ;       DLE  DC1  DC2  DC3  DC4  NAK  SYN  ETB          ;ASCII
        DB      010H,011H,012H,013H,03CH,03DH,032H,026H         ;EBCDIC
        ;       CAN  EM   SUB  ESC  IFS  IGS  IRS  IUS          ;ASCII
        DB      018H,019H,03FH,027H,01CH,01DH,01EH,01FH         ;EBCDIC
        ;       SPACE !   "    #    $    %    &    '            ;ASCII
        DB      040H,05AH,07FH,07BH,05BH,06CH,050H,07DH         ;EBCDIC
        ;        (    )    *    +    ,    -    .    /           ;ASCII
        DB      04DH,05DH,07CH,04EH,06BH,060H,04BH,061H         ;EBCDIC
        ;        0    1    2    3    4    5    6    7           ;ASCII
        DB      0F0H,0F1H,0F2H,0F3H,0F4H,0F5H,0F6H,0F7H         ;EBCDIC
        ;        8    9    :    ;    <    =    >    ?           ;ASCII
        DB      0F8H,0F9H,07AH,05EH,04CH,07EH,06EH,06FH         ;EBCDIC
        ;        @    A    B    C    D    E    F    G           ;ASCII
        DB      0FCH,0C1H,0C2H,0C3H,0C4H,0C5H,0C6H,0C7H         ;EBCDIC
        ;        H    I    J    K    L    M    N    O           ;ASCII
        DB      0C8H,0C9H,0D1H,0D2H,0D3H,0D4H,0D5H,0D6H         ;EBCDIC
        ;        P    Q    R    S    T    U    V    W           ;ASCII
        DB      0D7H,0D8H,0D9H,0E2H,0E3H,0E4H,0E5H,0E6H         ;EBCDIC
        ;        X    Y    Z    [    \    ]    ^    _           ;ASCII
        DB      0E7H,0E8H,0E9H,040H,0E0H,040H,040H,06DH         ;EBCDIC
```

```
;         `    a    b    c    d    e    f    g           ;ASCII
DB      079H,081H,082H,083H,084H,085H,086H,087H          ;EBCDIC
;         h    i    j    k    l    m    n    o           ;ASCII
DB      088H,089H,091H,092H,093H,094H,095H,096H          ;EBCDIC
;         p    q    r    s    t    u    v    w           ;ASCII
DB      097H,098H,099H,0A2H,0A3H,0A4H,0A5H,0A6H          ;EBCDIC
;         x    y    z    {    |    }    ~    DEL         ;ASCII
DB      0A7H,0A8H,0A9H,0C0H,06AH,0D0H,0A1H,007H          ;EBCDIC
;                                                                 ;
;                                                                 ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                         ;
;                                                                 ;
;                                                                 ;

SC4H:
        ;ПРЕОБРАЗОВАТЬ КОД 'A' ИЗ ASCII В EBCDIC
        MVI     A,'A'               ;ASCII 'A'
        CALL    ASC2EB              ;EBCDIC 'A' = 0C1H

        ;ПРЕОБРАЗОВАТЬ КОД '1' ИЗ ASCII В EBCDIC
        MVI     A,'1'               ;ASCII '1'
        CALL    ASC2EB              ;EBCDIC '1' = 0F1H

        ;ПРЕОБРАЗОВАТЬ КОД 'a' ИЗ ASCII В EBCDIC
        MVI     A,'a'               ;ASCII 'a'
        CALL    ASC2EB              ;EBCDIC 'a' = 081H

        JMP     SC4H

        END
```

4I. ПРЕОБРАЗОВАНИЕ СИМВОЛА В КОДЕ EBCDIC В ЕГО ЭКВИВАЛЕНТ В КОДЕ ASCII (EB2ASC)

Символ в коде EBCDIC преобразуется в его эквивалент в коде ASCII.

Процедура. Используется простой поиск в таблице, при этом данные являются индексом, а адрес ASCII – базой. Символы EBCDIC, которые печатаются, но не имеют эквивалента ASCII, транслируются в пробел в коде ASCII (20_{16}); символы EBCDIC, которые не имеют эквивалента ASCII и не печатаются, транслируются в ASCII NUL (00_{16}).

Используемые регистры: AF, DE, HL.
Время выполнения: 48 тактов.
Размер программы: 9 байт плюс 256 байт для таблицы преобразований.
Память, необходимая для данных: отсутствует.

УСЛОВИЯ НА ВХОДЕ

Символ EBCDIC в регистре A.

УСЛОВИЯ НА ВЫХОДЕ

Эквивалент в коде ASCII в регистре A.

ПРИМЕРЫ

1. Данные: (A) = 85_{16} (e в коде EBCDIC).
   Результат: (A) = 65_{16} (e в коде ASCII).
2. Данные: (A) = $4E_{16}$ (+ в коде EBCDIC).
   Результат: (A) = $2B_{16}$ (+ в коде ASCII).

```
;                                                               ;
;                                                               ;
;                                                               ;
;                                                               ;
;       ЗАГОЛОВОК:      ПРЕОБРАЗОВАНИЕ СИМВОЛА В КОДЕ EBCDIC     ;
;                       В ЕГО ЭКВИВАЛЕНТ В КОДЕ ASCII            ;
;       ИМЯ:            EB2ASC                                   ;
;                                                               ;
;                                                               ;
;                                                               ;
;       НАЗНАЧЕНИЕ:     ПРЕОБРАЗУЕТ СИМВОЛ В КОДЕ EBCDIC В ЕГО   ;
;                       ЭКВИВАЛЕНТ В КОДЕ  ASCII                 ;
;                                                               ;
;       ВХОД:           РЕГИСТР A = СИМВОЛ В КОДЕ EBCDIC         ;
;                                                               ;
;       ВЫХОД:          РЕГИСТР A = СИМВОЛ В КОДЕ ASCII          ;
;                                                               ;
;                                                               ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ:    AF,DE,HL                       ;
;                                                               ;
;       ВРЕМЯ:          48 ТАКТОВ                                ;
;                                                               ;
;       РАЗМЕР:         ПРОГРАММА -   9 БАЙТ                     ;
;                       ДАННЫЕ    - 256 БАЙТ ДЛЯ ТАБЛИЦЫ         ;
;                                                               ;
;                                                               ;

EB2ASC:
        LXI     H,ASCII         ;ВЗЯТЬ БАЗОВЫЙ АДРЕС ТАБЛИЦЫ ASCII
        MOV     E,A             ;ИСПОЛЬЗОВАТЬ КОД EBCDIC КАК ИНДЕКС
        MVI     D,0
        DAD     D
        MOV     A,M             ;ВЗЯТЬ СИМВОЛ В КОДЕ ASCII
        RET

;ТАБЛИЦА СООТВЕТСТВИЯ СИМВОЛОВ В КОДЕ EBCDIC И ASCII
; СИМВОЛЫ EBCDIC, КОТОРЫЕ ПЕЧАТАЮТСЯ, НО НЕ ИМЕЮТ ЭКВИВАЛЕНТА В КОДЕ
; ASCII, ТРАНСЛИРУЮТСЯ В ПРОБЕЛ В КОДЕ ASCII (020H), А СИМВОЛЫ EBCDIC,
; КОТОРЫЕ НЕ ПЕЧАТАЮТСЯ И НЕ ИМЕЮТ ЭКВИВАЛЕНТА В КОДЕ ASCII,
; ТРАНСЛИРУЮТСЯ В NUL В КОДЕ ASCII (000H)
ASCII:
;       NUL  SOH  STX  ETX       HT        DEL              ;EBCDIC
DB      000H,001H,002H,003H,000H,009H,000H,07FH             ;ASCII
;                      VT   FF   CR   SO   SI               ;EBCDIC
DB      000H,000H,000H,00BH,00CH,00DH,00EH,00FH             ;ASCII
;       DLE  DC1  DC2  DC3            BS                    ;EBCDIC
DB      010H,011H,012H,013H,000H,000H,008H,000H             ;ASCII
;       CAN  EM             IFS  IGS  IRS  IUS              ;EBCDIC
DB      018H,019H,000H,000H,01CH,01DH,01EH,01FH             ;ASCII
;                                LF   ETB  ESC              ;EBCDIC
DB      000H,000H,000H,000H,000H,00AH,017H,01BH             ;ASCII
;                                ENQ  ACK  BEL              ;EBCDIC
DB      000H,000H,000H,000H,000H,005H,006H,007H             ;ASCII
;                 SYN                      EOT              ;EBCDIC
DB      000H,000H,016H,000H,000H,000H,000H,004H             ;ASCII
;                           DC4  NAK       SUB              ;EBCDIC
DB      000H,000H,000H,000H,014H,015H,000H,01AH             ;ASCII
;       SPACE                                               ;EBCDIC
DB      ' ' ,000H,000H,000H,000H,000H,000H,000H             ;ASCII
;                 ¢    .    <    (    +    |                ;EBCDIC
DB      000H,000H,' ' ,'.' ,'<' ,'(' ,'+' ,' '              ;ASCII

;       &                                                   ;EBCDIC
DB      '&' ,000H,000H,000H,000H,000H,000H,000H             ;ASCII
;                 !    ¤    *    )    ;    ¬                ;EBCDIC
DB      000H,000H,'!' ,'¤' ,'*' ,')' ,';' ,' '              ;ASCII
```


6 Зак. 2265

```
;        _      /                                                    ;EBCDIC
DB      '_' ,'/' ,000H,000H,000H,000H,000H,000H                     ;ASCII
;                    |    ,    %    -    >    ?                      ;EBCDIC
DB      000H,000H,'|' ,',' ,'%' ,'-' ,'>' ,'?'                      ;ASCII
;                                                                    ;EBCDIC
DB      000H,000H,000H,000H,000H,000H,000H,000H                     ;ASCII
;             `    :    #    @    '    =    "                        ;EBCDIC
DB      000H,'`' ,':' ,'#' ,'@' ,'''' ,'=' ,'"'                     ;ASCII
;             a    b    c    d    e    f    g                        ;EBCDIC
DB      000H,'a' ,'b' ,'c' ,'d' ,'e' ,'f' ,'g'                      ;ASCII
;        h    i                                                      ;EBCDIC
DB      'h' ,'i' ,000H,000H,000H,000H,000H,000H                     ;ASCII
;             j    k    l    m    n    o    p                        ;EBCDIC
DB      000H,'j' ,'k' ,'l' ,'m' ,'n' ,'o' ,'p'                      ;ASCII
;        q    r                                                      ;EBCDIC
DB      'q' ,'r' ,000H,000H,000H,000H,000H,000H                     ;ASCII
;             ~    s    t    u    v    w    x                        ;EBCDIC
DB      000H,'~' ,'s' ,'t' ,'u' ,'v' ,'w' ,'x'                      ;ASCII
;        y    z                                                      ;EBCDIC
DB      'y' ,'z' ,000H,000H,000H,000H,000H,000H                     ;ASCII
;                                                                    ;EBCDIC
DB      000H,000H,000H,000H,000H,000H,000H,000H                     ;ASCII
;                                                                    ;EBCDIC
DB      000H,000H,000H,000H,000H,000H,000H,000H                     ;ASCII
;        {    A    B    C    D    E    F    G                        ;EBCDIC
DB      '{' ,'A' ,'B' ,'C' ,'D' ,'E' ,'F' ,'G'                      ;ASCII
;        H    I                                                      ;EBCDIC
DB      'H' ,'I' ,000H,000H,000H,000H,000H,000H                     ASCII
;        }    J    K    L    M    N    O    P                        ;EBCDIC
DB      '}' ,'J' ,'K' ,'L' ,'M' ,'N' ,'O' ,'P'                      ;ASCII
;        Q    R                                                      ;EBCDIC
DB      'Q' ,'R' ,000H,000H,000H,000H,000H,000H                     ;ASCII
;        \         S    T    U    V    W    X                        ;EBCDIC
DB      '\' ,000H,'S' ,'T' ,'U' ,'V' ,'W' ,'X'                      ;ASCII
;        Y    Z                                                      ;EBCDIC
DB      Y' ,'Z' ,000H,000H,000H,000H,000H,000H                      ;ASCII
;        0    1    2    3    4    5    6    7                        ;EBCDIC
DB      '0' ,'1' ,'2' ,'3' ,'4' ,'5' ,'6' ,'7'                      ;ASCII
;        8    9                                                      ;EBCDIC
DB      '8' ,'9' ,000H,000H,000H,000H,000H,000H                     ;ASCII

;                                                                    ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                            ;
;                                                                    ;

SC4I:
        ;ПРЕОБРАЗОВАТЬ КОД 'A' ИЗ EBCDIC В ASCII
        MVI     A,0C1H          ;EBCDIC 'A'
        CALL    EB2ASC          ;ASCII 'A' = 041H

        ;ПРЕОБРАЗОВАТЬ КОД '1' ИЗ EBCDIC В ASCII
        MVI     A,0F1H          ;EBCDIC '1'
        CALL    EB2ASC          ;ASCII '1' = 031H

        ;ПРЕОБРАЗОВАТЬ КОД 'a' ИЗ EBCDIC В ASCII
        MVI     A,081H          ;EBCDIC 'a'
        CALL    EB2ASC          ;ASCII 'a' = 061H

        JMP     SC4I
        END
```

ГЛАВА 5

РАБОТА С МАССИВАМИ И ИНДЕКСИРОВАНИЕ

5А. ЗАПОЛНЕНИЕ ПАМЯТИ (MFILL)

Определенное значение помещается в каждый байт области памяти известного размера, начиная с заданного адреса.

Процедура. Просто заполняется область памяти данным значением по байту за один раз.

Используемые регистры: AF, C, DE, HL.
Время выполнения: приблизительно 36 тактов на 1 байт плюс 14 тактов (8080) или приблизительно 37 тактов на 1 байт плюс 11 тактов (8085).
Размер программы: 10 байт.
Память, необходимая для данных: отсутствует.
Специальные случаи:

1. Размер 0000_{16} интерпретируется как 10000_{16}. Следовательно, заполняются определенным значением 65 536 байт.

2. Если заполняемая область занята или используется самой программой, то это вызовет непредсказуемые результаты. Очевидно, что заполнение области стека требует особого внимания, так как здесь сохраняется адрес возврата.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес области памяти в регистрах H и L.
Размер области (число байтов) в регистрах D и E.
Значение, помещаемое в память, в регистре A.

УСЛОВИЯ НА ВЫХОДЕ

Область заданного размера, начиная с базового адреса, заполняется определенным значением. Таким образом, заполняемая область начинается с BASE и продолжается до BASE + SIZE − 1 (BASE – базовый адрес, а SIZE – размер области).

ПРИМЕРЫ

1. Данные: значение = FF_{16},
   размер области (в байтах) = 0380_{16},
   базовый адрес = $1AE0_{16}$.
   Результат: FF_{16} помещается в ячейки памяти с адресами от $1AE0_{16}$ до $1E5F_{16}$.
2. Данные: значение = 00_{16} (в 8080, 8085 код операции для NOP);
   размер области (в байтах) = $1C65_{16}$;
   базовый адрес = $E34C_{16}$.
   Результат: 00_{16} помещается в ячейки памяти с адресами от $E34C_{16}$ до $FFB0_{16}$.

```
;                                                          ;
;                                                          ;
;                                                          ;
;                                                          ;
;       ЗАГОЛОВОК:      ЗАПОЛНЕНИЕ ПАМЯТИ                  ;
;       ИМЯ:            MFILL                              ;
```

```
;                                                                    ;
;       НАЗНАЧЕНИЕ:     ЗАПОЛНИТЬ ОБЛАСТЬ ПАМЯТИ ЗНАЧЕНИЕМ           ;
;                                                                    ;
;       ВХОД:           РЕГИСТР H = СТАРШИЙ БАЙТ БАЗОВОГО АДРЕСА     ;
;                       РЕГИСТР L = МЛАДШИЙ БАЙТ БАЗОВОГО АДРЕСА     ;
;                       РЕГИСТР D = СТАРШИЙ БАЙТ РАЗМЕРА ОБЛАСТИ     ;
;                       РЕГИСТР E = МЛАДШИЙ БАЙТ РАЗМЕРА ОБЛАСТИ     ;
;                       РЕГИСТР A = ЗНАЧЕНИЕ, ЗАПОЛНЯЮЩЕЕ ОБЛАСТЬ ПАМЯТИ;
;                                                                    ;
;                       ЗАМЕЧАНИЕ: РАЗМЕР 0 ИНТЕРПРЕТИРУЕТСЯ КАК 65536 ;
;                                                                    ;
;       ВЫХОД:          ОБЛАСТЬ ПАМЯТИ, ЗАПОЛНЕННАЯ ЗНАЧЕНИЕМ        ;
;                                                                    ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ:  AF,C,DE,HL                           ;
;                                                                    ;
;       ВРЕМЯ:          8080 - 36 ТАКТОВ НА БАЙТ ПЛЮС 14 ТАКТОВ      ;
;                       8085 - 37 ТАКТОВ НА БАЙТ ПЛЮС 11 ТАКТОВ      ;
;                                                                    ;
;       РАЗМЕР:         ПРОГРАММА - 10 БАЙТ                          ;
;                       ДАННЫЕ    - ОТСУТСТВУЮТ                      ;
;                                                                    ;
;                                                                    ;

MFILL:
        MOV     C,A             ;C = ЗАПИСЫВАЕМОЕ ЗНАЧЕНИЕ
LOOP:
        MOV     M,C             ;ЗАПОЛНИТЬ ОДИН БАЙТ ЗНАЧЕНИЕМ
        INX     H               ;УКАЗАТЬ НА СЛЕДУЮЩИЙ БАЙТ
        DCX     D               ;УМЕНЬШИТЬ СЧЕТЧИК
        MOV     A,E
        ORA     D
        JNZ     LOOP            ;ПРОДОЛЖАТЬ, ПОКА СЧЕТЧИК НЕ БУДЕТ = 0
        RET


;                                                                    ;
;                                                                    ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                            ;
;                                                                    ;
;                                                                    ;

SC5A:
        ;ЗАПОЛНИТЬ ОТ BF1 ДО BF1+15 НУЛЯМИ
        LXI     H,BF1           ;БАЗОВЫЙ АДРЕС
        LXI     D,SIZE1         ;ЧИСЛО БАЙТОВ
        MVI     A,0             ;ЗАПОЛНЯЮЩЕЕ ЗНАЧЕНИЕ
        CALL    MFILL           ;ЗАПОЛНИТЬ ПАМЯТЬ

        ;ЗАПОЛНИТЬ ОТ BF2 ДО BF2+1999 ЗНАЧЕНИЕМ FF
        LXI     H,BF2           ;БАЗОВЫЙ АДРЕС
        LXI     D,SIZE2         ;ЧИСЛО БАЙТОВ
        MVI     A,0FFH          ;ЗАПОЛНЯЮЩЕЕ ЗНАЧЕНИЕ
        CALL    MFILL           ;ЗАПОЛНИТЬ ПАМЯТЬ

        JMP     SC5A

SIZE1   EQU     16              ;РАЗМЕР БУФЕРА 1 (ШЕСТНАДЦАТЕРИЧНОЕ
                                ; ЧИСЛО 10)
```

```
SIZE2   EQU     2000            ;РАЗМЕР БУФЕРА 2 (ШЕСТНАДЦАТЕРИЧНОЕ
                                ; ЧИСЛО 07D0)
BF1:    DS      SIZE1           ;БУФЕР 1
BF2:    DS      SIZE2           ;БУФЕР 2

        END
```

5В. ПЕРЕСЫЛКА БЛОКА (BLKMOV)

Пересылает блок данных из исходной области в область назначения.

Процедура. Определяется, находится ли адрес области назначения внутри исходной области. Если это так, то при работе с начального адреса могли бы затеряться некоторые исходные данные. Во избежание этого пересылка выполняется вниз, начиная с самого верхнего адреса (этот прием иногда называется пересылкой справа). Если базовый адрес области назначения не лежит внутри исходной области, то просто пересылаются данные, начиная с самого младшего адреса (это иногда называют пересылкой слева). Размер области (число пересылаемых байтов) 0000_{16} вызывает выход из программы без изменения памяти. Программой автоматически обеспечивается циклический переход с конца памяти на начало по модулю 64К.

Используемые регистры: все.

Время выполнения: 48 тактов на байт (8080) или 50 тактов на байт (8085) плюс 97 тактов (8080) или 96 тактов (8085), если данные могут быть пересланы, начиная с младшего адреса, или же плюс 134 такта (8080) или 141 такт (8085), если данные должны пересылаться, начиная с самого старшего адреса (т. е. справа) в связи с перекрытием областей памяти.

Размер программы: 53 байта.

Память, необходимаи для данных: отсутствует.

Специальные случаи:

1. Размер (число байтов, которое должно быть переслано) 0 вызывает немедленный выход без изменения памяти.

2. Пересылка данных в область или из области, занятой или используемой самой программой или стеком, может привести к непредсказуемым результатам.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес исходной области в регистрах H и L.
Базовый адрес области назначения в регистрах D и E.
Число пересылаемых байтов в регистрах B и C.

УСЛОВИЯ НА ВЫХОДЕ

Блок памяти пересылается из исходной области в область назначения. Если число пересылаемых байтов равно NBYTES, DEST – базовый адрес области назначения, а SOURCE – базовый адрес исходной области, то данные с адреса SOURCE по SOURCE + NBYTES – 1 пересылаются по адресам с DEST по DEST + NBYTES – 1.

ПРИМЕРЫ

1. Данные: число пересылаемых байтов = 0200_{16},
базовый адрес области назначения = $05D1_{16}$,
базовый адрес исходной области = $035E_{16}$.

Результат: содержимое ячеек памяти $035E_{16}$ по $055D_{16}$ пересылается в ячейки памяти $05D1_{16}$ по $07D0_{16}$.

2. Данные: число пересылаемых байтов = $1B7A_{16}$, базовый адрес области назначения = $C946_{16}$, базовый адрес исходной области = $C300_{16}$.

Результат: содержимое ячеек памяти $C300_{16}$ по $DE79_{16}$ пересылается в ячейки памяти $C946_{16}$ по $E4BF_{16}$.

Заметим, что задача в примере 2 труднее, чем в примере 1, так как исходная область и область назначения перекрываются. Если бы программа просто пересылала данные в область назначения, начиная с младшего адреса, она могла бы переслать сначала содержимое ячейки памяти $C300_{16}$ в $C946_{16}$. При этом затерялось бы старое значение ячейки $C946_{16}$, которое необходимо позже переслать. Решение этой проблемы состоит в пересылке данных, начиная с самого старшего адреса, в тех случаях, когда область назначения начинается после исходной области, но перекрывает ее.

```
;                                                              ;
;                                                              ;
;                                                              ;
;                                                              ;
;      ЗАГОЛОВОК:      ПЕРЕСЫЛКА БЛОКА                         ;
;      ИМЯ:            BLKMOV                                  ;
;                                                              ;
;                                                              ;
;                                                              ;
;                                                              ;
;      НАЗНАЧЕНИЕ:     ПЕРЕСЛАТЬ ДАННЫЕ ИЗ ИСХОДНОГО БУФЕРА В БУФЕР ;
;                      НАЗНАЧЕНИЯ                              ;
;                                                              ;
;      ВХОД:           РЕГИСТР H = СТАРШИЙ БАЙТ АДРЕСА ИСХОДНОГО БУФЕРА;
;                      РЕГИСТР L = МЛАДШИЙ БАЙТ АДРЕСА ИСХОДНОГО БУФЕРА;
;                      РЕГИСТР D = СТАРШИЙ БАЙТ АДРЕСА БУФЕРА  ;
;                                  НАЗНАЧЕНИЯ                  ;
;                      РЕГИСТР E = МЛАДШИЙ БАЙТ АДРЕСА БУФЕРА  ;
;                                  НАЗНАЧЕНИЯ                  ;
;                      РЕГИСТР B = СТАРШИЙ БАЙТ ЧИСЛА ПЕРЕСЫЛАЕМЫХ ;
;                                  БАЙТОВ                      ;
;                      РЕГИСТР C = МЛАДШИЙ БАЙТ ЧИСЛА ПЕРЕСЫЛАЕМЫХ ;
;                                  БАЙТОВ                      ;
;                                                              ;
;      ВЫХОД:          ДАННЫЕ, ПЕРЕСЛАННЫЕ ИЗ ИСХОДНОГО БУФЕРА ;
;                      В БУФЕР НАЗНАЧЕНИЯ                      ;
;                                                              ;
;      ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ                              ;
;                                                              ;
;      ВРЕМЯ:          8080 - 48 ТАКТОВ НА БАЙТ ПЛЮС 101 ТАКТ, ;
;                             ЕСЛИ АДРЕС БУФЕРА НАЗНАЧЕНИЯ НЕ ПОПАДАЕТ ;
;                             В ИСХОДНЫЙ БУФЕР, ИЛИ ПЛЮС 157 ТАКТОВ В ;
;                             ПРОТИВНОМ СЛУЧАЕ                 ;
;                      8085 - 50 ТАКТОВ НА БАЙТ ПЛЮС 96 ТАКТОВ, ;
;                             ЕСЛИ АДРЕС БУФЕРА НАЗНАЧЕНИЯ НЕ ПОПАДАЕТ ;
;                             В ИСХОДНЫЙ БУФЕР, ИЛИ ПЛЮС 141 ТАКТ В ;
;                             ПРОТИВНОМ СЛУЧАЕ                 ;
;                                                              ;
;      РАЗМЕР:    ПРОГРАММА - 53 БАЙТА                         ;
```

```
;
;

BLKMOV:
        MOV     A,B             ;РАЗМЕР ПЕРЕСЫЛАЕМОГО БУФЕРА РАВЕН 0?
        ORA     C
        RZ                      ;ДА, ВОЗВРАТИТЬСЯ, НИЧЕГО НЕ ПЕРЕСЫЛАЯ

        ;ОПРЕДЕЛИТЬ, НАЧИНАЕТСЯ ЛИ БУФЕР НАЗНАЧЕНИЯ ВЫШЕ ИСХОДНОГО И
        ; ПЕРЕКРЫВАЕТ ЕГО (ПЕРЕКРЫТИЕ МОЖЕТ БЫТЬ ПО МОДУЛЮ 64K).
        ; ПЕРЕКРЫТИЕ ПОЛУЧАЕТСЯ В ТОМ СЛУЧАЕ, ЕСЛИ НАЧАЛЬНЫЙ АДРЕС
        ; БУФЕРА НАЗНАЧЕНИЯ МИНУС НАЧАЛЬНЫЙ АДРЕС ИСХОДНОГО БУФЕРА (ПО
        ; МОДУЛЮ 64K) МЕНЬШЕ ЧИСЛА ПЕРЕСЫЛАЕМЫХ БАЙТОВ
        PUSH    H               ;СОХРАНИТЬ АДРЕС ИСХОДНОГО БУФЕРА
        MOV     A,E             ;ВЫЧИСЛИТЬ РАЗНОСТЬ АДРЕСОВ ИСХОДНОГО
        SUB     L               ; БУФЕРА И БУФЕРА НАЗНАЧЕНИЯ
        MOV     L,A             ;СОХРАНИТЬ МЛАДШИЙ БАЙТ РАЗНОСТИ
        MOV     A,D
        SBB     H
        MOV     H,A             ;СОХРАНИТЬ СТАРШИЙ БАЙТ РАЗНОСТИ
        MOV     A,L             ;СРАВНИТЬ С ЧИСЛОМ ПЕРЕСЫЛАЕМЫХ БАЙТОВ
        SUB     C
        MOV     A,H
        SBB     B
        POP     H               ;ВОССТАНОВИТЬ АДРЕС ИСХОДНОГО БУФЕРА

        JNC     DOLEFT          ;ПЕРЕЙТИ, ЕСЛИ НЕТ ПЕРЕКРЫТИЯ

        ;БУФЕР НАЗНАЧЕНИЯ НАЧИНАЕТСЯ ПОСЛЕ ИСХОДНОГО БУФЕРА И
        ; ПЕРЕКРЫВАЕТ ЕГО
        ;ПЕРЕСЛАТЬ ИЗ СТАРШЕГО АДРЕСА ВО ИЗБЕЖАНИЕ ПОТЕРИ ДАННЫХ
        MOV     A,L             ;УКАЗАТЕЛЬ В ИСХОДНОМ БУФЕРЕ =
        ADD     C               ; УКАЗАТЕЛЬ В ИСХОДНОМ БУФЕРЕ + ДЛИНА
        MOV     L,A
        MOV     A,H
        ADC     B
        MOV     H,A
        MOV     A,E             ;УКАЗАТЕЛЬ В БУФЕРЕ НАЗНАЧЕНИЯ =
        ADD     C               ; УКАЗАТЕЛЬ В БУФЕРЕ НАЗНАЧЕНИЯ + ДЛИНА
        MOV     E,A
        MOV     A,D
        ADC     B
        MOV     D,A

        ;ПЕРЕСЛАТЬ БЛОК, НАЧИНАЯ СО СТАРШЕГО АДРЕСА
RHTLP:
        DCX     H               ;УМЕНЬШИТЬ УКАЗАТЕЛЬ В ИСХОДНОМ БУФЕРЕ
        DCX     D               ;УМЕНЬШИТЬ УКАЗАТЕЛЬ В БУФЕРЕ НАЗНАЧЕНИЯ
        MOV     A,M             ;ВЗЯТЬ СЛЕДУЮЩИЙ БАЙТ ИЗ ИСХОДНОГО БУФЕРА
        STAX    D               ;ПЕРЕСЛАТЬ ЕГО В БУФЕР НАЗНАЧЕНИЯ
        DCX     B               ;УМЕНЬШИТЬ СЧЕТЧИК
        MOV     A,B
        ORA     C
        JNZ     RHTLP           ;ПРОДОЛЖАТЬ, ПОКА СЧЕТЧИК НЕ СТАНЕТ = 0
        RET

        ;ПЕРЕСЛАТЬ ОБЫЧНЫМ ОБРАЗОМ, НАЧИНАЯ С МЛАДШЕГО АДРЕСА
```

```
DOLEFT:
        MOV     A,M                     ;ВЗЯТЬ СЛЕДУЮЩИЙ БАЙТ ИЗ ИСХОДНОГО БУФЕРА
        STAX    D                       ;ПЕРЕСЛАТЬ ЕГО В БУФЕР НАЗНАЧЕНИЯ
        INX     H                       ;УВЕЛИЧИТЬ АДРЕС ИСХОДНОГО БУФЕРА
        INX     D                       ;УВЕЛИЧИТЬ АДРЕС БУФЕРА НАЗНАЧЕНИЯ
        DCX     B                       ;УМЕНЬШИТЬ СЧЕТЧИК
        MOV     A,B
        ORA     C
        JNZ     DOLEFT                  ;ПРОДОЛЖАТЬ, ПОКА СЧЕТЧИК НЕ СТАНЕТ = 0

EXIT:
        RET



;                                                                              ;
;                                                                              ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                                      ;
;                                                                              ;
;                                                                              ;

SOURCE  EQU     2000H                   ;БАЗОВЫЙ АДРЕС ИСХОДНОГО БУФЕРА
DEST    EQU     2010H                   ;БАЗОВЫЙ АДРЕС БУФЕРА НАЗНАЧЕНИЯ
LEN     EQU     11H                     ;ЧИСЛО ПЕРЕСЫЛАЕМЫХ БАЙТОВ

        ;ПЕРЕСЛАТЬ 11 (ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО) БАЙТОВ ИЗ 2000H-2010H
        ; В 2010H-2020H
SC5B:
        LXI     H,SOURCE
        LXI     D,DEST
        LXI     B,LEN
        CALL    BLKMOV                  ;ПЕРЕСЛАТЬ ДАННЫЕ ИЗ ИСХОДНОГО БУФЕРА
                                        ; В БУФЕР НАЗНАЧЕНИЯ

        JMP     SC5B

        END
```

5С. ИНДЕКСИРОВАНИЕ ДВУМЕРНОГО МАССИВА БАЙТОВ (D2BYTE)

Вычисляется адрес элемента двумерного массива байтов, заданного базовым адресом массива, двумя индексами элемента и размером строки (т. е. числом столбцов). При этом считается, что массив хранится в памяти по строкам и что оба индекса начинаются с 0.

Процедура. Умножается размер строки (т. е. число столбцов в строке) на индекс строки (так как элементы хранятся по строкам) и прибавляется это произведение к индексу столбца. Затем полученная сумма прибавляется к базовому адресу. Умножение выполняется с использованием стандартного алгоритма сдвигов и сложения (см. подпрограмму 6В).

Используемые регистры: все.
Время выполнения: приблизительно 1500 тактов, что зависит главным образом от времени, необходимого для выполнения умножения.
Размер программы: 48 байт.
Память, необходимая для данных: 4 байта в любом месте памяти для хранения адреса возврата (2 байта с начальным адресом RETADR) и индекса столбца (2 байта с начальным адресом SS2).

УСЛОВИЯ НА ВХОДЕ

Порядок в стеке (начиная с вершины):
Младший байт адреса возврата.
Старший байт адреса возврата.
Младший байт индекса столбца.
Старший байт индекса столбца.
Младший байт размера строки (в байтах).
Старший байт размера строки (в байтах).
Младший байт индекса строки.
Старший байт индекса строки.
Младший байт базового адреса массива.
Старший байт базового адреса массива.

УСЛОВИЯ НА ВЫХОДЕ

Адрес элемента в регистрах H и L.

ПРИМЕРЫ

1. Данные: базовый адрес = $3C00_{16}$,
индекс столбца = 0004_{16},
размер строки (число столбцов) = 0018_{16},
индекс строки = 0003_{16}.
Результат: адрес элемента = $3C00_{16} + 0003_{16} * 0018_{16} + 0004_{16} =$
$= 3C00_{16} + 0048_{16} + 0004_{16} = 3C4C_{16}$.
Таким образом, адрес элемента ARRAY (3, 4) равен $3C4C_{16}$.
2. Данные: базовый адрес = $6A4A_{16}$,
индекс столбца = 0035_{16},
размер строки (число столбцов) = 0050_{16},
индекс строки = 0002_{16}.
Результат: адрес элемента = $6A4A_{16} + 0002_{16} * 0050_{16} + 0035_{16} =$
$= 6A4A_{16} + 00A0_{16} + 0035_{16} = 6B1F_{16}$.
Таким образом, адрес элемента ARRAY (2, 35) равен $6B1F_{16}$.

Заметим, что все индексы являются шестнадцатеричными ($35_{16} = 53_{10}$). Общая формула имеет вид

АДРЕС ЭЛЕМЕНТА = БАЗОВОМУ АДРЕСУ МАССИВА +
+ ИНДЕКС СТРОКИ * РАЗМЕР СТРОКИ + ИНДЕКС СТОЛБЦА

Обращаем внимание, что мы ссылаемся на *размер* индекса строки; этот размер равен числу последовательных адресов памяти, которые имеют одно и то же значение индекса. Это также и число байтов от начального адреса элемента до начального адреса элемента с тем же самым индексом столбца, но с индексом строки на единицу больше.

```
;                                                                   ;
;                                                                   ;
;                                                                   ;
;                                                                   ;
;       ЗАГОЛОВОК:      ИНДЕКСИРОВАНИЕ ДВУМЕРНОГО МАССИВА БАЙТОВ    ;
;       ИМЯ:            D2BYTE                                      ;
;                                                                   ;
;                                                                   ;
```

```
;                                                              ;
;       НАЗНАЧЕНИЕ:     ВЫЧИСЛЯЕТ АДРЕС A[I,J] ПО ЗАДАННОМУ БАЗОВОМУ   ;
;                       АДРЕСУ МАССИВА БАЙТОВ, ДВУМ ИНДЕКСАМ 'I' И 'J' ;
;                       И РАЗМЕРУ В БАЙТАХ ПО ПЕРВОМУ ИНДЕКСУ.         ;
;                       СЧИТАЕТСЯ, ЧТО МАССИВ ЗАПИСАН В ПАМЯТИ ПО      ;
;                       СТРОКАМ В ПОРЯДКЕ (A[0,0], A[0,1],..., A[K,L]), ;
;                       И ОБЕ РАЗМЕРНОСТИ НАЧИНАЮТСЯ С НУЛЯ, КАК В     ;
;                       СЛЕДУЮЩЕМ ПРЕДЛОЖЕНИИ НА ЯЗЫКЕ ПАСКАЛЬ:        ;
;                         A:ARRAY[0..2,0..7] OF BYTE;                  ;
;                                                              ;
;       ВХОД:           ВЕРШИНА СТЕКА                                  ;
;                         МЛАДШИЙ БАЙТ АДРЕСА ВОЗВРАТА                 ;
;                         СТАРШИЙ БАЙТ АДРЕСА ВОЗВРАТА                 ;
;                         МЛАДШИЙ БАЙТ ВТОРОГО ИНДЕКСА (ЭЛЕМЕНТ СТОЛБЦА);
;                         СТАРШИЙ БАЙТ ВТОРОГО ИНДЕКСА (ЭЛЕМЕНТ СТОЛБЦА);
;                         МЛАДШИЙ БАЙТ РАЗМЕРА ПО ПЕРВОМУ ИНДЕКСУ      ;
;                           В БАЙТАХ                                   ;
;                         СТАРШИЙ БАЙТ РАЗМЕРА ПО ПЕРВОМУ ИНДЕКСУ      ;
;                           В БАЙТАХ                                   ;
;                         МЛАДШИЙ БАЙТ РАЗМЕРА ИНДЕКСА ПЕРВОГО         ;
;                           (ЭЛЕМЕНТ СТРОКИ)                           ;
;                         СТАРШИЙ БАЙТ РАЗМЕРА ИНДЕКСА ПЕРВОГО         ;
;                           (ЭЛЕМЕНТ СТРОКИ)                           ;
;                         МЛАДШИЙ БАЙТ БАЗОВОГО АДРЕСА МАССИВА         ;
;                         СТАРШИЙ БАЙТ БАЗОВОГО АДРЕСА МАССИВА         ;
;                       ПРИМЕЧАНИЕ:                                    ;
;                         РАЗМЕР ПО ПЕРВОМУ ИНДЕКСУ РАВЕН ДЛИНЕ СТРОКИ ;
;                         В БАЙТАХ                                     ;
;                                                              ;
;       ВЫХОД:          РЕГИСТР H = СТАРШИЙ БАЙТ АДРЕСА ЭЛЕМЕНТА       ;
;                       РЕГИСТР L = МЛАДШИЙ БАЙТ АДРЕСА ЭЛЕМЕНТА       ;
;                                                              ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ:  ВСЕ                                    ;
;                                                              ;
;       ВРЕМЯ:          ПРИБЛИЗИТЕЛЬНО 1500 ТАКТОВ                     ;
;                                                              ;
;       РАЗМЕР:         ПРОГРАММА - 48 БАЙТ                            ;
;                       ДАННЫЕ    -  4 БАЙТА                           ;
;                                                              ;
;                                                              ;

D2BYTE:
        ;СОХРАНИТЬ АДРЕС ВОЗВРАТА
        POP     H
        SHLD    RETADR

        ;ВЗЯТЬ ВТОРОЙ ИНДЕКС
        POP     H
        SHLD    SS2

        ;ВЗЯТЬ РАЗМЕР ПО ПЕРВОМУ ИНДЕКСУ (ДЛИНУ СТРОКИ) И ПЕРВЫЙ ИНДЕКС
        POP     D               ;ВЗЯТЬ ДЛИНУ СТРОКИ
        POP     B               ;ВЗЯТЬ ПЕРВЫЙ ИНДЕКС

        ;УМНОЖИТЬ ПЕРВЫЙ ИНДЕКС НА ДЛИНУ СТРОКИ, ИСПОЛЬЗУЯ АЛГОРИТМ
        ; СДВИГА И СЛОЖЕНИЯ; ПРОИЗВЕДЕНИЕ ПОМЕСТИТЬ В HL
        LXI     H,0             ;ПРОИЗВЕДЕНИЕ = 0
        MVI     A,15            ;СЧЕТЧИК = ЧИСЛО РАЗРЯДОВ - 1
```

```
MLP:
        PUSH    PSW             ;СОХРАНИТЬ СЧЕТЧИК
        ORA     D               ;ФЛАГ ЗНАКА = СТАРШИЙ БАЙТ МНОЖИТЕЛЯ
                                ; (РАЗРЯД 7 СЧЕТЧИКА ВСЕГДА РАВЕН НУЛЮ)
        JP      MLP1            ;ПЕРЕЙТИ, ЕСЛИ СТАРШИЙ БАЙТ МНОЖИТЕЛЯ = 0
        DAD     B               ;ДОБАВИТЬ К ЧАСТИЧНОМУ ПРОИЗВЕДЕНИЮ
                                ; МНОЖИМОЕ
MLP1:   DAD     H               ;СДВИНУТЬ ЧАСТИЧНОЕ ПРОИЗВЕДЕНИЕ
        XCHG
        DAD     H               ;СДВИНУТЬ МНОЖИТЕЛЬ
        XCHG
        POP     PSW             ;ВОССТАНОВИТЬ СЧЕТЧИК
        DCR     A
        JNZ     MLP             ;ПРОДОЛЖАТЬ ДЛЯ 15 РАЗРЯДОВ

        ;ДОБАВИТЬ ПОСЛЕДНИЙ РАЗ, ЕСЛИ СТАРШИЙ БАЙТ МНОЖИТЕЛЯ РАВЕН 1
        ORA     D               ;ФЛАГ ЗНАКА = СТАРШИЙ РАЗРЯД МНОЖИТЕЛЯ
        JP      MLP2
        DAD     B               ;ДОБАВИТЬ МНОЖИМОЕ, ЕСЛИ ЗНАК РАВЕН 1

        ;ДОБАВИТЬ ВТОРОЙ ИНДЕКС
MLP2:   XCHG
        LHLD    SS2
        DAD     D

        ;ДЛЯ ФОРМИРОВАНИЯ ОКОНЧАТЕЛЬНОГО АДРЕСА ДОБАВИТЬ БАЗОВЫЙ АДРЕС
        POP     D               ;ВЗЯТЬ БАЗОВЫЙ АДРЕС МАССИВА
        DAD     D               ;ДОБАВИТЬ ЕГО К ИНДЕКСУ

        ;ВОЗВРАТИТЬСЯ В ВЫЗВАВШУЮ ПРОГРАММУ
        PUSH    H               ;ВОССТАНОВИТЬ В СТЕКЕ АДРЕС ВОЗВРАТА
        LHLD    RETADR
        XTHL
        RET

        ;ДАННЫЕ
RETADR: DS      2               ;ВРЕМЕННЫЕ ЯЧЕЙКИ ДЛЯ АДРЕСА ВОЗВРАТА
SS2:    DS      2               ;ВРЕМЕННЫЕ ЯЧЕЙКИ ДЛЯ ВТОРОГО ИНДЕКСА

;                                                                       ;
;                                                                       ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                               ;
;                                                                       ;
;                                                                       ;

SC5C:
        LXI     H,ARY           ;ПОМЕСТИТЬ В СТЕК БАЗОВЫЙ АДРЕС МАССИВА
        PUSH    H
        LHLD    SUBS1           ;ПОМЕСТИТЬ В СТЕК ПЕРВЫЙ ИНДЕКС
        PUSH    H
        LHLD    SSUBS1          ;ПОМЕСТИТЬ В СТЕК РАЗМЕР ПО ПЕРВОМУ
        PUSH    H               ; ИНДЕКСУ
        LHLD    SUBS2           ;ПОМЕСТИТЬ В СТЕК ВТОРОЙ ИНДЕКС
        PUSH    H
        CALL    D2BYTE          ;ВЫЧИСЛИТЬ АДРЕС ДЛЯ ИСХОДНЫХ ДАННЫХ
                                ; ТЕСТА
                                ; HL = АДРЕС ARY(2,4)
                                ;    = ARY + (2*8) + 4
```

```
                                       ;     = ARY + 20 (СОДЕРЖИТ 21)
                                       ;ЗАМЕТИМ, ЧТО ОБА ИНДЕКСА НАЧИНАЮТСЯ С 0

        JMP     SC5C

        ;ДАННЫЕ
SUBS1:  DW      2                      ;ИНДЕКС 1
SSUBS1: DW      8                      ;РАЗМЕР ПО ИНДЕКСУ 1
SUBS2:  DW      4                      ;ИНДЕКС 2

;МАССИВ (3 СТРОКИ ПО 8 КОЛОНОК)
ARY:    DB      1 ,2 ,3 ,4 ,5 ,6 ,7 ,8
        DB      9 ,10,11,12,13,14,15,16
        DB      17,18,19,20,21,22,23,24

        END
```

5D. ИНДЕКСИРОВАНИЕ ДВУМЕРНОГО МАССИВА СЛОВ (D2WORD)

Вычисляется начальный адрес элемента двумерного массива слов (слово имеет длину 16 разрядов), заданного базовым адресом массива, двумя индексами элемента и размером строки в байтах. Считается, что массив хранится по строкам и оба индекса начинаются с 0.

Процедура. Умножается размер строки (в байтах) на индекс строки (так как элементы хранятся по строкам), прибавляется произведение к удвоенному индексу столбца (удвоенному, так как каждый элемент занимает два байта), а затем эта сумма прибавляется к базовому адресу. В программе используется стандартный алгоритм сдвигов и сложения (см. подпрограмму 6B) для умножения.

Используемые регистры: все.
Время выполнения: приблизительно 1500 тактов, что зависит главным образом от времени, необходимого для умножения размера строки на индекс строки.
Размер программы: 49 байт.
Память, необходимая для данных: 4 байта в любом месте памяти для хранения адреса возврата (2 байта с начальным адресом RETADR) и индекса столбца (2 байта с начальным адресом SS2).

УСЛОВИЯ НА ВХОДЕ

Порядок в стеке (начиная с вершины):
Младший байт адреса возврата.
Старший байт адреса возврата.
Младший байт индекса столбца.
Старший байт индекса етолбца.
Младший байт размера строки (в байтах).
Старший байт размера строки (в байтах).
Младший байт индекса строки.
Старший байт индекса строки.
Младший байт базового адреса массива.
Старший байт базового адреса массива.

УСЛОВИЯ НА ВЫХОДЕ

Начальный адрес элемента в регистрах H и L.
Элемент занимает адрес в регистрах H и L и следующий больший адрес.

ПРИМЕРЫ

1. Данные: базовый адрес = $5E14_{16}$,
   индекс столбца = 0008_{16},
   размер строки (в байтах) = $001C_{16}$ (т. е. каждая строка имеет 0014_{10} или $000E_{16}$ элементов длиной в слово),
   индекс строки = 0005_{16}.

   Результат: начальный адрес элемента = $5E14_{16} + 0005_{16} * 001C_{16} + 0008_{16} * 2 =$
   $= 5E14_{16} + 0008C_{16} + 0010_{16} = 5EB0_{16}$.

   Таким образом, адрес элемента ARRAY (5,8) равен $5EB0_{16}$ и элемент занимает $5EB0_{16}$ и $5EB1_{16}$.

2. Данные: базовый адрес = $B100_{16}$,
   индекс столбца = 0002_{16},
   размер строки (в байтах) = 0008_{16} (т. е. каждая строка содержит четыре элемента длиной в слово),
   индекс строки = 0006_{16}.

   Результат: начальный адрес элемента = $B100_{16} + 0006_{16} * 0008_{16} + 0002_{16} * 2 =$
   $= B100_{16} + 0030_{16} + 0004_{16} = B134_{16}$.

   Таким образом, адрес элемента ARRAY (6, 2) равен $B134_{16}$, и элемент занимает $B134_{16}$ и $B135_{16}$.

Общая формула имеет вид:

НАЧАЛЬНЫЙ АДРЕС ЭЛЕМЕНТА = БАЗОВОМУ АДРЕСУ МАССИВА +
+ ИНДЕКС СТРОКИ * РАЗМЕР СТРОКИ + ИНДЕКС СТОЛБЦА * 2

Заметим, что одним из параметров этой подпрограммы является размер строки в байтах. Для элементов длиной в слово размером является число столбцов в строке, умноженное на 2 (размер элемента в байтах). Причина того, что был выбран именно этот параметр, а не число столбцов в строке или максимальный индекс столбца, заключается в том, что этот параметр может быть вычислен один раз (когда определяются границы массива) и может использоваться повсюду, где производится доступ к массиву. Другие параметры (число столбцов или максимальный индекс столбца) могли бы потребовать дополнительных вычислений при каждой операции индексации.

```
;                                                                    ;
;                                                                    ;
;                                                                    ;
;                                                                    ;
;       ЗАГОЛОВОК:      ИНДЕКСИРОВАНИЕ ДВУМЕРНОГО МАССИВА СЛОВ       ;
;       ИМЯ:            D2WORD                                       ;
;                                                                    ;
;                                                                    ;


;                                                                    ;
;       НАЗНАЧЕНИЕ:     ВЫЧИСЛЯЕТ АДРЕС A[I,J] ПО ЗАДАННОМУ БАЗОВОМУ ;
;                       АДРЕСУ МАССИВА СЛОВ, ДВУМ ИНДЕКСАМ 'I' И 'J' ;
;                       И РАЗМЕРУ В БАЙТАХ ПО ПЕРВОМУ ИНДЕКСУ.       ;
;                       СЧИТАЕТСЯ, ЧТО МАССИВ ЗАПИСАН В ПАМЯТИ ПО    ;
;                       СТРОКАМ В ПОРЯДКЕ (A[0,0], A[0,1],..., A[K,L]), ;
;                       И ОБЕ РАЗМЕРНОСТИ НАЧИНАЮТСЯ С НУЛЯ, КАК В   ;
```

```
;                       СЛЕДУЮЩЕМ ПРЕДЛОЖЕНИИ НА ЯЗЫКЕ ПАСКАЛЬ:         ;
;                         A:ARRAY[0..2,0..7] OF WORD;                   ;
;                                                                       ;
;       ВХОД:           ВЕРШИНА СТЕКА                                   ;
;                         МЛАДШИЙ БАЙТ АДРЕСА ВОЗВРАТА                  ;
;                         СТАРШИЙ БАЙТ АДРЕСА ВОЗВРАТА                  ;
;                         МЛАДШИЙ БАЙТ ВТОРОГО ИНДЕКСА (ЭЛЕМЕНТ СТОЛБЦА);
;                         СТАРШИЙ БАЙТ ВТОРОГО ИНДЕКСА (ЭЛЕМЕНТ СТОЛБЦА);
;                         МЛАДШИЙ БАЙТ РАЗМЕРА ПО ПЕРВОМУ ИНДЕКСУ       ;
;                           В БАЙТАХ                                    ;
;                         СТАРШИЙ БАЙТ РАЗМЕРА ПО ПЕРВОМУ ИНДЕКСУ       ;
;                           В БАЙТАХ                                    ;
;                         МЛАДШИЙ БАЙТ ПЕРВОГО ИНДЕКСА (ЭЛЕМЕНТ СТРОКИ) ;
;                         СТАРШИЙ БАЙТ ПЕРВОГО ИНДЕКСА (ЭЛЕМЕНТ СТРОКИ) ;
;                         МЛАДШИЙ БАЙТ БАЗОВОГО АДРЕСА МАССИВА          ;
;                         СТАРШИЙ БАЙТ БАЗОВОГО АДРЕСА МАССИВА          ;
;                       ПРИМЕЧАНИЕ:                                     ;
;                         РАЗМЕР ПО ПЕРВОМУ ИНДЕКСУ РАВЕН ДЛИНЕ СТРОКИ  ;
;                         В СЛОВАХ *2                               ;
;                                                                       ;
;       ВЫХОД:          РЕГИСТР H = СТАРШИЙ БАЙТ АДРЕСА ЭЛЕМЕНТА        ;
;                       РЕГИСТР L = МЛАДШИЙ БАЙТ АДРЕСА ЭЛЕМЕНТА        ;
;                                                                       ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ:  ВСЕ                                     ;
;                                                                       ;
;       ВРЕМЯ:          ПРИБЛИЗИТЕЛЬНО 1500 ТАКТОВ                      ;
;                                                                       ;
;       РАЗМЕР:         ПРОГРАММА - 49 БАЙТ                             ;
;                       ДАННЫЕ    -  4 БАЙТА                            ;
;                                                                       ;
;                                                                       ;

D2WORD:
        ;СОХРАНИТЬ АДРЕС ВОЗВРАТА
        POP     H
        SHLD    RETADR

        ;ВЗЯТЬ ВТОРОЙ ИНДЕКС, УМНОЖИТЬ НА 2 ДЛЯ ЭЛЕМЕНТОВ ДЛИНОЙ В СЛОВО
        POP     H
        DAD     H               ;*2
        SHLD    SS2

        ;ВЗЯТЬ РАЗМЕР ПО ПЕРВОМУ ИНДЕКСУ (ДЛИНУ СТРОКИ) И ПЕРВЫЙ ИНДЕКС
        POP     D               ;ВЗЯТЬ ДЛИНУ СТРОКИ
        POP     B               ;ВЗЯТЬ ПЕРВЫЙ ИНДЕКС

        ;УМНОЖИТЬ ПЕРВЫЙ ИНДЕКС НА ДЛИНУ СТРОКИ, ИСПОЛЬЗУЯ АЛГОРИТМ
        ; СДВИГА И СЛОЖЕНИЯ; ПРОИЗВЕДЕНИЕ ПОМЕСТИТЬ В HL
        LXI     H,0             ;ПРОИЗВЕДЕНИЕ = 0
        MVI     A,15            ;СЧЕТЧИК = ЧИСЛО РАЗРЯДОВ - 1
MLP:
        PUSH    PSW             ;СОХРАНИТЬ СЧЕТЧИК
        ORA     D               ;ФЛАГ ЗНАКА = СТАРШИЙ БАЙТ МНОЖИТЕЛЯ
                                ; (РАЗРЯД 7 ВСЕГДА РАВЕН НУЛЮ)
        JP      MLP1            ;ПЕРЕЙТИ, ЕСЛИ СТАРШИЙ БАЙТ МНОЖИТЕЛЯ =
        DAD     B               ;ДОБАВИТЬ К ЧАСТИЧНОМУ ПРОИЗВЕДЕНИЮ
                                ; МНОЖИМОЕ
```

```
MLP1:   DAD     H               ;СДВИНУТЬ ЧАСТИЧНОЕ ПРОИЗВЕДЕНИЕ
        XCHG
        DAD     H               ;СДВИНУТЬ МНОЖИТЕЛЬ
        XCHG
        POP     PSW             ;ВОССТАНОВИТЬ СЧЕТЧИК
        DCR     A
        JNZ     MLP             ;ПРОДОЛЖАТЬ ДЛЯ 15 РАЗРЯДОВ

        ;ДОБАВИТЬ МНОЖИМОЕ ПОСЛЕДНИЙ РАЗ, ЕСЛИ СТАРШИЙ БАЙТ МНОЖИТЕЛЯ
        ; РАВЕН 1
        ORA     D               ;ФЛАГ ЗНАКА = СТАРШИЙ РАЗРЯД МНОЖИТЕЛЯ
        JP      MLP2
        DAD     B               ;ДОБАВИТЬ МНОЖИМОЕ, ЕСЛИ ЗНАК РАВЕН 1

        ;ДОБАВИТЬ ВТОРОЙ ИНДЕКС
MLP2:   XCHG
        LHLD    SS2
        DAD     D

        ;ДЛЯ ФОРМИРОВАНИЯ ОКОНЧАТЕЛЬНОГО АДРЕСА ДОБАВИТЬ БАЗОВЫЙ АДРЕС
        POP     D               ;ВЗЯТЬ БАЗОВЫЙ АДРЕС МАССИВА
        DAD     D               ;ДОБАВИТЬ ЕГО К ИНДЕКСУ

        ;ВОЗВРАТИТЬСЯ В ВЫЗВАВШУЮ ПРОГРАММУ
        PUSH    H               ;ВОССТАНОВИТЬ В СТЕКЕ АДРЕС ВОЗВРАТА
        LHLD    RETADR
        XTHL
        RET

        ;ДАННЫЕ
RETADR: DS      2               ;ВРЕМЕННЫЕ ЯЧЕЙКИ ДЛЯ АДРЕСА ВОЗВРАТА
SS2:    DS      2               ;ВРЕМЕННЫЕ ЯЧЕЙКИ ДЛЯ ВТОРОГО ИНДЕКСА

;                                                               ;
;                                                               ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                       ;
;                                                               ;
;                                                               ;

SC5D:
        LXI     H,ARY           ;ПОМЕСТИТЬ В СТЕК БАЗОВЫЙ АДРЕС МАССИВА
        PUSH    H
        LHLD    SUBS1           ;ПОМЕСТИТЬ В СТЕК ПЕРВЫЙ ИНДЕКС
        PUSH    H
        LHLD    SSUBS1          ;ПОМЕСТИТЬ В СТЕК РАЗМЕР ПО ПЕРВОМУ
        PUSH    H               ; ИНДЕКСУ
        LHLD    SUBS2           ;ПОМЕСТИТЬ В СТЕК ВТОРОЙ ИНДЕКС
        PUSH    H
        CALL    D2WORD          ;ВЫЧИСЛИТЬ АДРЕС ДЛЯ ИСХОДНЫХ ДАННЫХ
                                ; ТЕСТА:
                                ; HL = АДРЕС ARY(2,4)
                                ;    = ARY + (2*16) + 4 * 2
                                ;    = ARY + 40 (СОДЕРЖИМОЕ РАВНО 2100H)
                                ;ЗАМЕТИМ, ЧТО ОБА ИНДЕКСА НАЧИНАЮТСЯ С 0

       JMP     SC5D
```

```
        ;ДАННЫЕ
SUBS1:  DW      2                   ;ИНДЕКС 1
SSUBS1: DW      16                  ;РАЗМЕР ПО ИНДЕКСУ 1
SUBS2:  DW      4                   ;ИНДЕКС 2

;МАССИВ (3 СТРОКИ ПО 8 КОЛОНОК)
ARY:    DW      0100H,0200H,0300H,0400H,0500H,0600H,0700H,0800H
        DW      0900H,1000H,1100H,1200H,1300H,1400H,1500H,1600H
        DW      1700H,1800H,1900H,2000H,2100H,2200H,2300H,2400H

        END
```

5E. ИНДЕКСИРОВАНИЕ N-МЕРНОГО МАССИВА (NDIM)

Вычисляется начальный адрес элемента N-мерного массива, заданного базовым адресом и N парами размеров и индексов. Размер каждой размерности равен числу байтов от начального адреса элемента до начального адреса элемента с индексом в данной размерности на единицу больше, но с теми же самыми индексами во всех других размерностях. Считается, что массив хранится по строкам (т. е. массив организован таким образом, что индексы справа изменяются быстрее, чем индексы, стоящие от них слева).

Заметим, что размер самого правого индекса является просто размером элемента (в байтах); размер следующего индекса равен размеру элемента, умноженному на максимальное значение самого правого индекса плюс 1, и так далее. Кроме того, считается, что все индексы начинаются с 0. Иначе говоря, пользователь должен нормализовать индексы. (См. второй пример в конце листинга.)

Процедура. Для каждой размерности в цикле вычисляется смещение данной размерности, равное индексу, умноженному на размер. В простейшем случае, если размер является степенью двух, умножение заменяется на сдвиги влево. В противном случае выполняется умножение с помощью алгоритма сдвигов и сложения, описанного в подпрограмме 6В. После вычисления всех смещений, они прибавляются к базовому адресу и таким образом получается начальный адрес элемента.

Используемые регистры: все.
Время выполнения: приблизительно 1700 тактов на размерность плюс дополнительно 170 тактов (что зависит главным образом от того, сколько раз необходимо выполнить умножение).
Размер программы: 130 байт.
Память, необходимая для данных: 5 байт в любом месте памяти для хранения адреса возврата (2 байта с начальным адресом RETADR), накопленного смещения (2 байта с начальным адресом OFFSET) и числа размерностей (1 байт с адресом NUMDIM).
Специальный случай: если число разностей равно 0, то осуществляется возврат с базовым адресом в регистры H и L.

УСЛОВИЯ НА ВХОДЕ

Порядок в стеке (начиная с вершины):
Младший байт адреса возврата.

Старший байт адреса возврата.
Младший байт числа размерностей.
Старший байт числа размерностей (не используется).
Младший байт размера самой правой размерности.
Старший байт размера самой правой размерности.
Младший байт самого правого индекса.
Старший байт самого правого индекса.
.
.
.
Младший байт размера самой левой размерности.
Старший байт размера самой левой размерности.
Младший байт самого левого индекса.
Старший байт самого левого индекса.
Младший байт базового адреса массива.
Старший байт базового адреса массива.

УСЛОВИЯ НА ВЫХОДЕ

Начальный адрес элемента в регистрах H и L.
Элемент занимает память с адресами от START до START + SIZE – 1, где START – вычисленный адрес, SIZE – размер элемента в байтах.

ПРИМЕРЫ

1. Данные: базовый адрес = $3C00_{16}$,
число размерностей = 03_{16},
правый индекс = 0005_{16},
правый размер = 0003_{16} (запись длиной 3 байта),
средний индекс = 0003_{16},
средний размер = 0012_{16} (шесть 3-байтных записей),
левый индекс = 0004_{16},
левый размер = $007E_{16}$ (7 наборов по шесть 3-байтных записей).

Результат: начальный адрес элемента = $3C00_{16} + 0005_{16} * 0003_{16} + 0003_{16} * 0012_{16} + 0004_{16} * 007E_{16} = 3C00_{16} + 000F_{16} + 0036_{16} + 01F8_{16} = 3E3D_{16}$.

Таким элементом является элемент ARRAY (4,3,5); он занимает адреса $3E3D_{16}$ ... ... $3E3F_{16}$. Максимальные значения индексов равны 6 (для левого) и 5 (для среднего), при этом каждый элемент занимает три байта.

Общая формула имеет вид:

$$\text{НАЧАЛЬНЫЙ АДРЕС} = \text{БАЗОВЫЙ АДРЕС} \sum_{i=0}^{N-1} \text{ИНДЕКС}_i * \text{РАЗМЕР}_i,$$

где

N – число размерностей; ИНДЕКС_i – i-й индекс; РАЗМЕР_i – размер i-й размерности.

Заметим, что размер каждой размерности используется в качестве параметра, что позволяет уменьшить число повторяющихся умножений и придать процедуре бо́льшую общность. Когда границы массива становятся известными, эти размеры могут быть вычислены и сохранены. Они могут использоваться каждый раз при индексировании массива. Очевидно, что если границы массива фиксированы, то эти размеры не изменяются и не должны вычис-

ляться заново при каждой операции индексирования. Кроме того, эти размеры являются общими, так как элементы могут сами состоять из любого числа байтов:

```
;                                                                        ;
;                                                                        ;
;                                                                        ;
;                                                                        ;
;       ЗАГОЛОВОК:      ИНДЕКСИРОВАНИЕ N-МЕРНОГО МАССИВА                 ;
;       ИМЯ:            NDIM                                             ;
;                                                                        ;
;                                                                        ;


;                                                                        ;
;       НАЗНАЧЕНИЕ:     ВЫЧИСЛЯЕТ АДРЕС ЭЛЕМЕНТА N-МЕРНОГО МАССИВА       ;
;                       ПО ЗАДАННОМУ БАЗОВОМУ АДРЕСУ, N ПАРАМ            ;
;                       ДЛИН В БАЙТАХ И ИНДЕКСОВ, А ТАКЖЕ ЧИСЛУ          ;
;                       РАЗМЕРНОСТЕЙ МАССИВА. СЧИТАЕТСЯ, ЧТО МАССИВ      ;
;                       ЗАПИСАН В ПАМЯТИ ПО СТРОКАМ В ПОРЯДКЕ            ;
;                       (A[0,0,0],A[0,0,1],...,A[0,1,0],A[0,1,1],...).   ;
;                       КРОМЕ ТОГО, СЧИТАЕТСЯ, ЧТО ВСЕ РАЗМЕРНОСТИ       ;
;                       НАЧИНАЮТСЯ С НУЛЯ, КАК В СЛЕДУЮЩЕМ ПРЕДЛОЖЕНИИ   ;
;                       НА ЯЗЫКЕ ПАСКАЛЬ:                                ;
;                         A:ARRAY[0..10,0..3,0..5] OF SOMETHING          ;
;                                                                        ;
;       ВХОД:           ВЕРШИНА СТЕКА                                    ;
;                         МЛАДШИЙ БАЙТ АДРЕСА ВОЗВРАТА                   ;
;                         СТАРШИЙ БАЙТ АДРЕСА ВОЗВРАТА                   ;
;                         МЛАДШИЙ БАЙТ ЧИСЛА РАЗМЕРНОСТЕЙ                ;
;                         СТАРШИЙ БАЙТ ЧИСЛА РАЗМЕРНОСТЕЙ                ;
;                           (НЕ ИСПОЛЬЗУЕТСЯ)                            ;
;                         МЛАДШИЙ БАЙТ ДЛИНЫ РАЗМЕРНОСТИ N-1             ;
;                         СТАРШИЙ БАЙТ ДЛИНЫ РАЗМЕРНОСТИ N-1             ;
;                         МЛАДШИЙ БАЙТ ИНДЕКСА РАЗМЕРНОСТИ N-1           ;
;                         СТАРШИЙ БАЙТ ИНДЕКСА РАЗМЕРНОСТИ N-1           ;
;                         МЛАДШИЙ БАЙТ ДЛИНЫ РАЗМЕРНОСТИ N-2             ;
;                         СТАРШИЙ БАЙТ ДЛИНЫ РАЗМЕРНОСТИ N-2             ;
;                         МЛАДШИЙ БАЙТ ИНДЕКСА РАЗМЕРНОСТИ N-2           ;
;                         СТАРШИЙ БАЙТ ИНДЕКСА РАЗМЕРНОСТИ N-2           ;
;                               .                                        ;
;                               .                                        ;
;                               .                                        ;
;                         МЛАДШИЙ БАЙТ ДЛИНЫ РАЗМЕРНОСТИ 0               ;
;                         СТАРШИЙ БАЙТ ДЛИНЫ РАЗМЕРНОСТИ 0               ;
;                         МЛАДШИЙ БАЙТ ИНДЕКСА РАЗМЕРНОСТИ 0             ;
;                         СТАРШИЙ БАЙТ ИНДЕКСА РАЗМЕРНОСТИ 0             ;
;                         МЛАДШИЙ БАЙТ БАЗОВОГО АДРЕСА МАССИВА           ;
;                         СТАРШИЙ БАЙТ БАЗОВОГО АДРЕСА МАССИВА           ;
;                       ПРИМЕЧАНИЕ:                                      ;
;                         ВСЕ ДЛИНЫ РАЗМЕРНОСТЕЙ В БАЙТАХ                ;
;                                                                        ;
;       ВЫХОД:          РЕГИСТР H = СТАРШИЙ БАЙТ НАЧАЛЬНОГО АДРЕСА       ;
;                                   ЭЛЕМЕНТА                             ;
;                       РЕГИСТР L = МЛАДШИЙ БАЙТ НАЧАЛЬНОГО АДРЕСА       ;
;                                   ЭЛЕМЕНТА                             ;
;                                                                        ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ:  ВСЕ                                      ;
;                                                                        ;
```

```
;       ВРЕМЯ:          ПРИБЛИЗИТЕЛЬНО 1700 ТАКТОВ НА РАЗМЕРНОСТЬ      ;
;                       ПЛЮС 170 ТАКТОВ                                ;
;                                                                      ;
;       РАЗМЕР:         ПРОГРАММА - 130 БАЙТ                           ;
;                       ДАННЫЕ    -   5 БАЙТ                           ;
;                                                                      ;

NDIM:
        ;ВЗЯТЬ ПАРАМЕТРЫ ИЗ СТЕКА
        POP     H               ;СОХРАНИТЬ АДРЕС ВОЗВРАТА
        SHLD    RETADR

        ;СМЕЩЕНИЕ := 0
        LXI     H,0
        SHLD    OFFSET

        ;ВЗЯТЬ ЧИСЛО РАЗМЕРНОСТЕЙ И ПРОВЕРИТЬ НА 0
        POP     H
        MOV     A,L
        STA     NUMDIM          ;ВЗЯТЬ ЧИСЛО РАЗМЕРНОСТЕЙ
        ORA     A               ;ПРОВЕРИТЬ НА 0
        JZ      ADBASE          ;ЕСЛИ НЕТ РАЗМЕРНОСТЕЙ, ТО ВЕРНУТЬСЯ
                                ; В ВЫЗЫВАЮЩУЮ ПРОГРАММУ С БАЗОВЫМ
                                ; АДРЕСОМ В HL

        ;ЦИКЛ ДЛЯ КАЖДОЙ РАЗМЕРНОСТИ
        ; ВЫЧИСЛЯЕТСЯ СМЕЩЕНИЕ := СМЕЩЕНИЕ + (ИНДЕКС * РАЗМЕР)
LOOP:
        POP     D               ;ВЗЯТЬ РАЗМЕР
        POP     H               ;ВЗЯТЬ ИНДЕКС
        CALL    NXTOFF          ;СМЕЩЕНИЕ := СМЕЩЕНИЕ + (ИНДЕКС * РАЗМЕР)
        LXI     H,NUMDIM
        DCR     M               ;УМЕНЬШИТЬ ЧИСЛО РАЗМЕРНОСТЕЙ
        JNZ     LOOP            ;ПРОДОЛЖАТЬ ДЛЯ ВСЕХ РАЗМЕРНОСТЕЙ

ADBASE:
        ;ВЫЧИСЛИТЬ НАЧАЛЬНЫЙ АДРЕС ЭЛЕМЕНТА
        ;СМЕЩЕНИЕ := БАЗА + СМЕЩЕНИЕ
        LHLD    OFFSET
        POP     D               ;ВЗЯТЬ БАЗОВЫЙ АДРЕС
        DAD     D               ;СЛОЖИТЬ СО СМЕЩЕНИЕМ

        ;ВОССТАНОВИТЬ АДРЕС ВОЗВРАТА И ВЫЙТИ
        XCHG
        LHLD    RETADR          ;ВОССТАНОВИТЬ В СТЕКЕ АДРЕС ВОЗВРАТА
        PUSH    H
        XCHG
        RET

        ;--------------------------------------
        ;ПОДПРОГРАММА NXTOFF
        ;НАЗНАЧЕНИЕ: СМЕЩЕНИЕ := СМЕЩЕНИЕ + (ИНДЕКС * РАЗМЕР)
        ;ВХОД:  СМЕЩЕНИЕ = ТЕКУЩЕЕ ЗНАЧЕНИЕ СМЕЩЕНИЯ
        ;       DE = ТЕКУЩИЙ РАЗМЕР ДАННОЙ РАЗМЕРНОСТИ
        ;       HL = ТЕКУЩИЙ ИНДЕКС
        ;ВЫХОД: СМЕЩЕНИЕ = СМЕЩЕНИЕ + (ИНДЕКС * РАЗМЕР)
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ
        ;--------------------------------------
```

```
NXTOFF:
        PUSH    H               ;СОХРАНИТЬ В СТЕКЕ ТЕКУШИЙ ИНДЕКС

        ;ПРОВЕРИТЬ, ЯВЛЯЕТСЯ ЛИ РАЗМЕР СТЕПЕНЬЮ 2 И МЕНЬШЕ,
        ; ЧЕМ 256
        MOV     A,D
        ORA     A               ;СТАРШИЙ БАЙТ РАВЕН 0?

        JNZ     BIGSZ           ;ПЕРЕЙТИ, ЕСЛИ РАЗМЕР БОЛЬШЕ 256

        MOV     A,E             ;A = МЛАДШИЙ БАЙТ РАЗМЕРА
        LXI     H,EASYAY        ;HL = БАЗОВЫЙ АДРЕС EASYAY
        MVI     B,SZEASY        ;B = РАЗМЕР МАССИВА EASYAY
        MVI     C,0             ;C = СЧЕТЧИК СДВИГОВ
EASYLP:
        CMP     M
        JZ      ISEASY          ;ПЕРЕЙТИ, ЕСЛИ РАЗМЕР ЯВЛЯЕТСЯ СТЕПЕНЬЮ 2
        INX     H               ;УВЕЛИЧИТЬ ДЛЯ СЛЕДУЮЩЕГО БАЙТА МАССИВА
                                ; EASYAY
        INR     C               ;УВЕЛИЧИТЬ СЧЕТЧИК СДВИГОВ
        DCR     B               ;УМЕНЬШИТЬ СЧЕТЧИК
        JNZ     EASYLP          ;ПЕРЕЙТИ, ЕСЛИ НЕ ВСЕ ЭЛЕМЕНТЫ МАССИВА
                                ; EASYAY
        JMP     BIGSZ           ;ПЕРЕЙТИ, ЕСЛИ РАЗМЕР НЕ ЯВЛЯЕТСЯ
                                ; СТЕПЕНЬЮ 2

ISEASY:
        POP     H               ;ВЗЯТЬ ИНДЕКС
        MOV     A,C             ;ВЗЯТЬ ЧИСЛО СДВИГОВ
        ORA     A               ;ПРОВЕРИТЬ НА 0
        JZ      ADDOFF          ;ПЕРЕЙТИ, ЕСЛИ ЧИСЛО СДВИГОВ = 0

        ;РАЗМЕР ЭЛЕМЕНТА * ИНДЕКС ВЫЧИСЛЯЕТСЯ С ПОМОЩЬЮ СДВИГОВ ВЛЕВО
SHIFT:
        DAD     H               ;УМНОЖИТЬ ИНДЕКС НА 2
        DCR     A               ;УМЕНЬШИТЬ СЧЕТЧИК СДВИГОВ
        JNZ     SHIFT           ;ПРОДОЛЖАТЬ ДО ВЫПОЛНЕНИЯ УМНОЖЕНИЯ
        JMP     ADDOFF          ;ПОСЛЕ УМНОЖЕНИЯ ПРИБАВИТЬ
                                ; СМЕЩЕНИЕ + ИНДЕКС

BIGSZ:
        ;ТАК КАК РАЗМЕР НЕ ЯВЛЯЕТСЯ СТЕПЕНЬЮ 2, ПРОИЗВЕСТИ УМНОЖЕНИЕ
        ; РАЗМЕРА ЭЛЕМЕНТА НА ИНДЕКС БОЛЕЕ СЛОЖНЫМ СПОСОБОМ
        POP     B               ;ВЗЯТЬ ИНДЕКС

        ;СНАЧАЛА УМНОЖИТЬ ИНДЕКС * ДЛИНА СТРОКИ, ИСПОЛЬЗУЯ АЛГОРИТМ
        ; СДВИГОВ И СЛОЖЕНИЯ. ПОМЕСТИТЬ ПРОИЗВЕДЕНИЕ В HL
        ; BC = ИНДЕКС (МНОЖИМОЕ)
        ; DE = РАЗМЕР (МНОЖИТЕЛЬ)
        LXI     H,0             ;ПРОИЗВЕДЕНИЕ = 0
        MVI     A,15            ;СЧЕТЧИК = ЧИСЛО РАЗРЯДОВ - 1
MLP:
        PUSH    PSW             ;СОХРАНИТЬ СЧЕТЧИК
        ORA     D               ;ФЛАГ ЗНАКА = СТАРШИЙ БАЙТ МНОЖИТЕЛЯ
                                ; (РАЗРЯД 7 СЧЕТЧИКА ВСЕГДА 0)
        JP      MLP1            ;ПЕРЕЙТИ, ЕСЛИ СТАРШИЙ БАЙТ МНОЖИТЕЛЯ = 0
        DAD     B               ;ДЛЯ ПОЛУЧЕНИЯ ЧАСТИЧНОГО ПРОИЗВЕДЕНИЯ
                                ; ПРИБАВИТЬ МНОЖИМОЕ
```

```
MLP1:   DAD     H               ;СДВИНУТЬ ЧАСТИЧНОЕ ПРОИЗВЕДЕНИЕ
        XCHG
        DAD     H               ;СДВИНУТЬ МНОЖИТЕЛЬ
        XCHG
        POP     PSW             ;ВОССТАНОВИТЬ СЧЕТЧИК
        DCR     A
        JNZ     MLP             ;ПРОДОЛЖАТЬ ДЛЯ 15 РАЗРЯДОВ

        ;ЕСЛИ СТАРШИЙ БАЙТ МНОЖИТЕЛЯ РАВЕН 1, ТО ПРИБАВИТЬ МНОЖИМОЕ
        ; ПОСЛЕДНИЙ РАЗ
        ORA     D               ;ФЛАГ ЗНАКА = СТАРШИЙ БАЙТ МНОЖИТЕЛЯ
        JP      ADDOFF
        DAD     B               ;ПРИБАВИТЬ МНОЖИМОЕ, ЕСЛИ ЗНАК = 1

        ;ДОБАВИТЬ К СМЕЩЕНИЮ ИНДЕКС * РАЗМЕР
ADDOFF:
        XCHG
        LHLD    OFFSET          ;ВЗЯТЬ СМЕЩЕНИЕ
        DAD     D               ;ДОБАВИТЬ ПРОИЗВЕДЕНИЕ ИНДЕКС * РАЗМЕР
        SHLD    OFFSET          ;СОХРАНИТЬ СМЕЩЕНИЕ
        RET

EASYAY:                         ;ЧИСЛО СДВИГОВ
        DB      1               ;0
        DB      2               ;1
        DB      4               ;2
        DB      8               ;3
        DB      16              ;4
        DB      32              ;5
        DB      64              ;6
        DB      128             ;7
SZEASY  EQU     *-EASYAY

        ;ДАННЫЕ
RETADR: DS      2               ;ВРЕМЕННЫЕ ЯЧЕЙКИ ДЛЯ АДРЕСА ВОЗВРАТА
OFFSET: DS      2               ;ВРЕМЕННЫЕ ЯЧЕЙКИ ДЛЯ ЧАСТИЧНОГО СМЕЩЕНИЯ
NUMDIM: DS      1               ;ЧИСЛО РАЗМЕРНОСТЕЙ

;                                                                       ;
;                                                                       ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                               ;
;                                                                       ;
;                                                                       ;

SC5E:
        ;НАЙТИ АДРЕС AY1[1,3,0]
        ; ТАК КАК НИЖНИЕ ПРЕДЕЛЫ МАССИВА 1 НУЛИ, НЕТ НЕОБХОДИМОСТИ
        ; ИХ НОРМИРОВАТЬ

        ;ПОМЕСТИТЬ В СТЕК БАЗОВЫЙ АДРЕС МАССИВА 1
        LXI     H,AY1
        PUSH    H

        ;ПОМЕСТИТЬ В СТЕК ИНДЕКС И РАЗМЕР ДЛЯ РАЗМЕРНОСТИ 1
        LXI     H,1             ;ИНДЕКС
        PUSH    H
        LXI     H,A1SZ1         ;РАЗМЕР
        PUSH    H
```

```
;ПОМЕСТИТЬ В СТЕК ИНДЕКС И РАЗМЕР ДЛЯ РАЗМЕРНОСТИ 2
LXI     H,3                 ;ИНДЕКС
PUSH    H
LXI     H,A1SZ2             ;РАЗМЕР
PUSH    H

;ПОМЕСТИТЬ В СТЕК ИНДЕКС И РАЗМЕР ДЛЯ РАЗМЕРНОСТИ 3
LXI     H,0                 ;ИНДЕКС
PUSH    H
LXI     H,A1SZ3             ;РАЗМЕР
PUSH    H

;ПОМЕСТИТЬ В СТЕК ЧИСЛО РАЗМЕРНОСТЕЙ
LXI     H,A1DIM             ;РАЗМЕРНОСТИ
PUSH    H

CALL    NDIM                ;ВЫЧИСЛИТЬ АДРЕС
                            ;AY = НАЧАЛЬНЫЙ АДРЕС ARY1(1,3,0)
                            ;    = ARY + (1*126) + (3*21) + (0*3)
                            ;    = ARY + 189
;ВЫЧИСЛИТЬ АДРЕС AY2[-1,6]
; ТАК КАК НИЖНИЕ ПРЕДЕЛЫ МАССИВА 2 НЕ НАЧИНАЮТСЯ С НУЛЯ,
; ИНДЕКСЫ ДОЛЖНЫ БЫТЬ НОРМИРОВАНЫ

;ПОМЕСТИТЬ В СТЕК БАЗОВЫЙ АДРЕС МАССИВА 2
LXI     H,AY2
PUSH    H

;ПОМЕСТИТЬ В СТЕК (ИНДЕКС - НИЖНИЙ ПРЕДЕЛ) И РАЗМЕР ДЛЯ РАЗМЕРНОСТИ 1
LXI     H,-1                ;ИНДЕКС
LXI     D,-A2D1L            ;ОТРИЦАТЕЛЬНЫЙ НИЖНИЙ ПРЕДЕЛ
DAD     D                   ;ПРИБАВИТЬ ОТРИЦАТЕЛЬНОЕ ЗНАЧЕНИЕ ДЛЯ
                            ; НОРМИРОВАНИЯ К 0
PUSH    H
LXI     H,A2SZ1             ;РАЗМЕР
PUSH    H

;ПОМЕСТИТЬ В СТЕК (ИНДЕКС - НИЖНИЙ ПРЕДЕЛ) И РАЗМЕР ДЛЯ РАЗМЕРНОСТИ 2
LXI     H,6                 ;ИНДЕКС
LXI     D,-A2D2L            ;ОТРИЦАТЕЛЬНЫЙ НИЖНИЙ ПРЕДЕЛ
DAD     D                   ;ПРИБАВИТЬ ОТРИЦАТЕЛЬНОЕ ЗНАЧЕНИЕ ДЛЯ
                            ; НОРМИРОВАНИЯ К 0
PUSH    H
LXI     H,A2SZ2             ;РАЗМЕР
PUSH    H

;ПОМЕСТИТЬ В СТЕК ЧИСЛО РАЗМЕРНОСТЕЙ
LXI     H,A2DIM
PUSH    H

CALL    NDIM                ;ВЫЧИСЛИТЬ АДРЕС
                            ;AY = АДРЕС ARY1(-1,6)
                            ;    = APY+(((-1)-(-5))*18)+((-6)*2)
                            ;    = ARY + 80

JMP     SC5E
```

```
;ДАННЫЕ
;AY1 : ARRAY[A1D1L..A1D1H,A1D2L..A1D2H,A1D3L..A1D3H] 3-BYTE ELEMENTS
;         [ 0  ..  3  ,  0  ..  5  ,  0  ..  6  ]
A1DIM   EQU    3                         ;ЧИСЛО РАЗМЕРНОСТЕЙ
A1D1L   EQU    0                         ;НИЖНЯЯ ГРАНИЦА РАЗМЕРНОСТИ 1
A1D1H   EQU    3                         ;ВЕРХНЯЯ ГРАНИЦА РАЗМЕРНОСТИ 1
A1D2L   EQU    0                         ;НИЖНЯЯ ГРАНИЦА РАЗМЕРНОСТИ 2
A1D2H   EQU    5                         ;ВЕРХНЯЯ ГРАНИЦА РАЗМЕРНОСТИ 2
A1D3L   EQU    0                         ;НИЖНЯЯ ГРАНИЦА РАЗМЕРНОСТИ 3
A1D3H   EQU    6                         ;ВЕРХНЯЯ ГРАНИЦА РАЗМЕРНОСТИ 3
A1SZ3   EQU    3                         ;РАЗМЕР ЭЛЕМЕНТА В РАЗМЕРНОСТИ 3
A1SZ2   EQU    ((A1D3H-A1D3L)+1)*A1SZ3   ;РАЗМЕР ЭЛЕМЕНТА В РАЗМЕРНОСТИ 2
A1SZ1   EQU    ((A1D2H-A1D2L)+1)*A1SZ2   ;РАЗМЕР ЭЛЕМЕНТА В РАЗМЕРНОСТИ 1
AY1:    DS     ((A1D1H-A1D1L)+1)*A1SZ1   ;МАССИВ
;AY2 : ARRAY[A2D1L..A2D1H,A2D2L..A2D2H] OF WORD
;         [ -5 ..  -1 ,  2  ..  10 ]
A2DIM   EQU    2                         ;ЧИСЛО РАЗМЕРНОСТЕЙ
A2D1L   EQU    -5                        ;НИЖНЯЯ ГРАНИЦА РАЗМЕРНОСТИ 1
A2D1H   EQU    -1                        ;ВЕРХНЯЯ ГРАНИЦА РАЗМЕРНОСТИ 1
A2D2L   EQU    2                         ;НИЖНЯЯ ГРАНИЦА РАЗМЕРНОСТИ 2
A2D2H   EQU    10                        ;ВЕРХНЯЯ ГРАНИЦА РАЗМЕРНОСТИ 2
A2SZ2   EQU    2                         ;РАЗМЕР ЭЛЕМЕНТА В РАЗМЕРНОСТИ 2
A2SZ1   EQU    ((A2D2H-A2D2L)+1)*A2SZ2   ;РАЗМЕР ЭЛЕМЕНТА В РАЗМЕРНОСТИ 1
AY2:    DS     ((A2D1H-A2D1L)+1)*A2SZ1   ;МАССИВ

        END
```

ГЛАВА 6

АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ

6А. ШЕСТНАДЦАТИРАЗРЯДНОЕ ВЫЧИТАНИЕ (SUB16)

Вычитаются два 16-разрядных числа. При этом флаг переноса действует как заем.

Процедура. Просто из уменьшаемого вычитается вычитаемое по одному байту за один раз, начиная с младших байтов. При вычитании старших байтов учитывается заем. При вычитании старших байтов устанавливаются флаги.

Используемые регистры: AF, DE, HL.
Время выполнения: 38 тактов (8080) или 34 такта (8085).
Размер программы: 7 байт.
Память, необходимая для данных: отсутствует.

УСЛОВИЯ НА ВХОДЕ

Уменьшаемое в регистрах H и L.
Вычитаемое в регистрах D и E.

УСЛОВИЯ НА ВЫХОДЕ

Разность в регистрах H и L.

1. Данные: уменьшаемое = $A45D_{16}$,
   вычитаемое = $97E1_{16}$.
   Результат: разность = $0C7C_{16}$,
   флаг переноса = 0 (нет заема).
2. Данные: уменьшаемое = $03E1_{16}$,
   вычитаемое = $07E4_{16}$.
   Результат: разность = $FBFD_{16}$,
   флаг переноса = 1 (есть заем).

```
;                                                                    ;
;                                                                    ;
;                                                                    ;
;                                                                    ;
;       ЗАГОЛОВОК:      16-РАЗРЯДНОЕ ВЫЧИТАНИЕ                       ;
;       ИМЯ:            SUB16                                        ;
;                                                                    ;
;                                                                    ;

;                                                                    ;
;       НАЗНАЧЕНИЕ:     ВЫЧИТАЕТ 16-РАЗРЯДНЫЕ ЗНАКОВЫЕ ИЛИ БЕЗЗНАКОВЫЕ ;
;                       СЛОВА С ВОЗВРАТОМ 16-РАЗРЯДНОЙ ЗНАКОВОЙ ИЛИ  ;
;                       БЕЗЗНАКОВОЙ РАЗНОСТИ                         ;
;                                                                    ;
;       ВХОД:           РЕГИСТР L = МЛАДШИЙ БАЙТ УМЕНЬШАЕМОГО        ;
;                       РЕГИСТР H = СТАРШИЙ БАЙТ УМЕНЬШАЕМОГО        ;
;                       РЕГИСТР E = МЛАДШИЙ БАЙТ ВЫЧИТАЕМОГО         ;
;                       РЕГИСТР D = СТАРШИЙ БАЙТ ВЫЧИТАЕМОГО         ;
;                                                                    ;
;       ВЫХОД:          РАЗНОСТЬ = УМЕНЬШАЕМОЕ - ВЫЧИТАЕМОЕ          ;
;                       РЕГИСТР L = МЛАДШИЙ БАЙТ РАЗНОСТИ            ;
;                       РЕГИСТР H = СТАРШИЙ БАЙТ РАЗНОСТИ            ;
;                                                                    ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF,DE,HL                              ;
;                                                                    ;
;       ВРЕМЯ:          8080 = 38 ТАКТОВ                             ;
;                       8085 = 34 ТАКТА                              ;
;                                                                    ;
;       РАЗМЕР:         ПРОГРАММА - 7 БАЙТ                           ;

SUB16:
        MOV     A,L             ;ВЫЧЕСТЬ МЛАДШИЙ БАЙТ
        SUB     E
        MOV     L,A
        MOV     A,H             ;ВЫЧЕСТЬ СТАРШИЙ БАЙТ С ЗАЕМОМ
        SBB     D
        MOV     H,A
        RET


;                                                                    ;
;                                                                    ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                            ;
;                                                                    ;
;                                                                    ;
```

```
SC6A:
        ;ВЫЧЕСТЬ 1023 ИЗ 123
        LXI     H,123           ;HL = УМЕНЬШАЕМОЕ
        LXI     D,1023          ;DE = ВЫЧИТАЕМОЕ
        CALL    SUB16           ;123 - 1023 = -900 = 0FC7CH
                                ; РЕГИСТР L = 7CH, РЕГИСТР H = 0FCH
                                ; ФЛАГ ПЕРЕНОСА = 1 (БЫЛ ЗАЕМ РАЗРЯДА)

        JMP     SC6A

        END
```

6В. ШЕСТНАДЦАТИРАЗРЯДНОЕ УМНОЖЕНИЕ (MUL16)

Умножаются два 16-разрядных операнда и возвращается младшее по значению слово (16-разрядное) произведения.

Процедура. Используется обычный алгоритм сдвигов и сложения, при котором множимое добавляется к частичному произведению каждый раз, когда в множителе находится единичный разряд. Для правильного относительного расположения операндов и произведения в программе 15 раз осуществляется сдвиг влево множителя и промежуточного произведения (т. е. на число разрядов в множителе минус 1). При этом старший (16-й) разряд произведения теряется.

Используемые регистры: все.
Время выполнения: приблизительно от 1003 до 1167 тактов (8080) или от 1001 до 1065 тактов (8085) в зависимости, главным образом, от числа единичных разрядов в множителе.
Размер программы: 26 байт.
Память, необходимая для данных: отсутствует.

УСЛОВИЯ НА ВХОДЕ

Множимое в регистрах H и L.
Множитель в регистрах D и E.

УСЛОВИЯ НА ВЫХОДЕ

Младшее по значению слово произведения в регистрах H и L.

ПРИМЕРЫ

1. Данные: множитель = 0012_{16},
   множимое = $03D1_{16}$.
   Результат: произведение = $44B2_{16}$,
   старшее по значению слово равно 0.
2. Данные: множитель = $37D1_{16}$,
   множимое = $A045_{16}$.
   Результат: произведение = $AB55_{16}$.

В действительности это младшее по значению 16-разрядное слово 32-разрядного произведения $22F1AB55_{16}$.

```
;                                                                     ;
;                                                                     ;
;                                                                     ;
;       ЗАГОЛОВОК:      16-РАЗРЯДНОЕ УМНОЖЕНИЕ                        ;
;       ИМЯ:            MUL16                                         ;
;                                                                     ;
;                                                                     ;


;                                                                     ;
;       НАЗНАЧЕНИЕ:     УМНОЖАЕТ 16-РАЗРЯДНЫЕ ЗНАКОВЫЕ ИЛИ БЕЗЗНАКОВЫЕ ;
;                       СЛОВА И ВОЗВРАЩАЕТ 16-РАЗРЯДНОЕ ЗНАКОВОЕ ИЛИ  ;
;                       БЕЗЗНАКОВОЕ ПРОИЗВЕДЕНИЕ
;                                                                     ;
;       ВХОД:           РЕГИСТР L = МЛАДШИЙ БАЙТ МНОЖИМОГО            ;
;                       РЕГИСТР H = СТАРШИЙ БАЙТ МНОЖИМОГО            ;
;                       РЕГИСТР E = МЛАДШИЙ БАЙТ МНОЖИТЕЛЯ            ;
;                       РЕГИСТР D = СТАРШИЙ БАЙТ МНОЖИТЕЛЯ            ;
;                                                                     ;
;       ВЫХОД:          ПРОИЗВЕДЕНИЕ := МНОЖИМОЕ * МНОЖИТЕЛЬ          ;
;                       РЕГИСТР L = МЛАДШИЙ БАЙТ ПРОИЗВЕДЕНИЯ         ;
;                       РЕГИСТР H = СТАРШИЙ БАЙТ ПРОИЗВЕДЕНИЯ         ;
;                                                                     ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ                                    ;
;                                                                     ;
;       ВРЕМЯ:          ПРИБЛИЗИТЕЛЬНО ОТ 1003 ДО 1167 ТАКТОВ ДЛЯ 8080 ;
;                       ПРИБЛИЗИТЕЛЬНО ОТ 1001 ДО 1065 ТАКТОВ ДЛЯ 8085 ;
;                                                                     ;
;       РАЗМЕР:         ПРОГРАММА - 26 БАЙТ                           ;
;                                                                     ;
;                                                                     ;

        ;ЗАДАТЬ НАЧАЛЬНЫЕ ЗНАЧЕНИЯ ПРОИЗВЕДЕНИЯ И СЧЕТЧИКА РАЗРЯДОВ
MUL16:
        MOV     C,L             ;ПЕРЕСЛАТЬ МНОЖИМОЕ В BC
        MOV     B,H
        LXI     H,0             ;ПРОИЗВЕДЕНИЕ = 0
        MVI     A,15            ;СЧЕТЧИК = ДЛИНА ОПЕРАНДА В РАЗРЯДАХ - 1

        ;АЛГОРИТМ СДВИГОВ И СЛОЖЕНИЯ
        ;ЕСЛИ СТАРШИЙ РАЗРЯД МНОЖИТЕЛЯ РАВЕН 1, ТО ДОБАВИТЬ МНОЖИМОЕ
        ; К ПРОМЕЖУТОЧНОМУ ПРОИЗВЕДЕНИЮ
        ;СДВИНУТЬ ПРОМЕЖУТОЧНОЕ ПРОИЗВЕДЕНИЕ И МНОЖИТЕЛЬ ВЛЕВО НА
        ; 1 РАЗРЯД
MLP:
        PUSH    PSW             ;СОХРАНИТЬ СЧЕТЧИК
        ORA     D               ;ФЛАГ ЗНАКА = СТАРШИЙ РАЗРЯД МНОЖИТЕЛЯ
                                ; (РАЗРЯД 7 СЧЕТЧИКА ВСЕГДА РАВЕН НУЛЮ)
        JP      MLP1            ;ПЕРЕЙТИ, ЕСЛИ СТАРШИЙ РАЗРЯД МНОЖИТЕЛЯ
                                ; РАВЕН 0
        DAD     B               ;ДОБАВИТЬ МНОЖИМОЕ К ПРОМЕЖУТОЧНОМУ
                                ; ПРОИЗВЕДЕНИЮ
MLP1:   DAD     H               ;СДВИНУТЬ ПРОМЕЖУТОЧНОЕ ПРОИЗВЕДЕНИЕ
        XCHG
        DAD     H               ;СДВИНУТЬ МНОЖИТЕЛЬ
        XCHG
        POP     PSW             ;ВОССТАНОВИТЬ СЧЕТЧИК
        DCR     A
```

```
        JNZ     MLP                     ;ПРОДОЛЖАТЬ, ПОКА СЧЕТЧИК НЕ БУДЕТ
                                        ; РАВЕН 0

        ;ЕСЛИ СТАРШИЙ РАЗРЯД МНОЖИТЕЛЯ РАВЕН 1, ДОБАВИТЬ МНОЖИМОЕ
        ; ПОСЛЕДНИЙ РАЗ
        ORA     D                       ;ФЛАГ ЗНАКА = СТАРШИЙ РАЗРЯД МНОЖИТЕЛЯ
        RP                              ;ВЫЙТИ ИЗ ПОДПРОГРАММЫ, ЕСЛИ СТАРШИЙ
                                        ; РАЗРЯД МНОЖИТЕЛЯ = 0
        DAD     B                       ;ИНАЧЕ ДОБАВИТЬ МНОЖИМОЕ К ПРОИЗВЕДЕНИЮ
        RET



;                                                                       ;
;                                                                       ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                               ;
;                                                                       ;
;                                                                       ;

SC6B:
        LXI     H,-2                    ;HL = МНОЖИМОЕ
        LXI     D,1023                  ;DE = МНОЖИТЕЛЬ
        CALL    MUL16                   ;УМНОЖИТЬ 16-РАЗРЯДНЫЕ ЧИСЛА
                                        ;РЕЗУЛЬТАТ УМНОЖЕНИЯ
                                        ; 1023 * -2 = -2046 = 0F802H
                                        ; РЕГИСТР L = 02H,
                                        ; РЕГИСТР H = F8H

        JMP     SC6B

        END
```

6С. ШЕСТНАДЦАТИРАЗРЯДНОЕ ДЕЛЕНИЕ (SDIV16, UDIV16)

Делятся два 16-разрядных операнда и возвращается частное и остаток. Имеются две входные точки: SDIV16 делит два 16-разрядных операнда со знаками, в то время как UDIV16 делит два 16-разрядных операнда без знаков. При делении на 0 флаг переноса устанавливается в 1, а частное и остаток равны 0; в противном случае флаг переноса очищается.

Процедура. Если операнды имеют знаки, то определяется знак частного и берутся абсолютные значения отрицательных операндов. Кроме того, должен сохраняться знак делимого, так как он определяет знак остатка. Затем с помощью алгоритма сдвигов и вычитания выполняется беззнаковое деление. Частное и делимое сдвигаются влево, при этом каждый раз, когда пробное вычитание было успешным, единичный разряд помещается в частное. Если операнды имели знаки, то программа должна превратить в отрицательное число (т. е. вычесть из 0) частное или остаток, если только они должны быть отрицательными. При делении без ошибок флаг переноса очищается, а при делении на 0 – устанавливается. Кроме того, если делитель равен 0, то частное и остаток также равны 0.

Используемые регистры: все.
Время выполнения: приблизительно от 2630 до 3200 тактов (8080) или от 2480 до 2950 тактов (8085), что зависит главным образом от того, сколько было ус-

пешных пробных вычитаний, потребовавших замещения предыдущего частного остатком.

Размер программы: 136 байт.

Память, необходимая для данных: 3 байта в любом месте ОЗУ для знака частного (адрес SQUOT), знака остатка (адрес SREM) и счетчика цикла деления (адрес COUNT).

Специальный случай: если делитель равен 0, программа возвращает флаг переноса, равный 1, при этом частное и остаток равны 0.

УСЛОВИЯ НА ВХОДЕ

Делимое в регистрах H и L.
Делитель в регистрах D и E.

УСЛОВИЯ НА ВЫХОДЕ

Частное в регистрах H и L.
Остаток в регистрах D и E.
Если делитель не равен 0, флаг переноса равен 0 и результат нормальный. Если делитель равен 0, флаг переноса равен 1, а частное и остаток равны 0000_{16}.

ПРИМЕРЫ

1. Данные: делимое = $03E0_{16}$,
   делитель = $00B6_{16}$.
   Результат: частное (от UDIV16) = 0005_{16},
   остаток (от UDIV16) = 0052_{16},
   флаг переноса = 0 (нет ошибки деления на 0).
2. Данные: делимое = $D73A_{16}$,
   делитель = $02F1_{16}$.
   Результат: частное (от SDIV16) = $FFF3_{16}$,
   остаток (от SDIV16) = $ED77_{16}$,
   флаг переноса = 0 (нет ошибки деления на 0).

Остаток от деления чисел со знаками может быть как положительным, так и отрицательным. В данной процедуре остаток всегда принимает знак делимого. Если принять, что остаток всегда положительный, то его легко получить из отрицательного остатка. Для этого надо вычесть 1 из частного и прибавить к остатку делитель. Тогда результат примера 2 будет равен:

частное = $FFF2_{16}$ = -14_{10},
остаток (всегда положительный) = 0068_{16}.

```
;                                                          ;
;                                                          ;
;                                                          ;
;                                                          ;
;       ЗАГОЛОВОК:      16-РАЗРЯДНОЕ ДЕЛЕНИЕ               ;
;       ИМЯ:            SDIV16, UDIV16                     ;
;                                                          ;
;                                                          ;


;                                                          ;
;       НАЗНАЧЕНИЕ:     SDIV16                             ;
;                         ДЕЛИТ ДВА 16-РАЗРЯДНЫХ СЛОВА СО ЗНАКОМ ;
;                         И ВОЗВРАЩАЕТ 16-РАЗРЯДНЫЕ СО ЗНАКОМ ЧАСТНОЕ ;
;                         И ОСТАТОК                        ;
```

```
;                       UDIV16                                               ;
;                         ДЕЛИТ ДВА 16-РАЗРЯДНЫХ СЛОВА БЕЗ ЗНАКА             ;
;                         И ВОЗВРАЩАЕТ 16-РАЗРЯДНЫЕ БЕЗ ЗНАКА ЧАСТНОЕ        ;
;                         И ОСТАТОК                                          ;
;                                                                            ;
;       ВХОД:           РЕГИСТР L = МЛАДШИЙ БАЙТ ДЕЛИМОГО                    ;
;                       РЕГИСТР H = СТАРШИЙ БАЙТ ДЕЛИМОГО                    ;
;                       РЕГИСТР E = МЛАДШИЙ БАЙТ ДЕЛИТЕЛЯ                    ;
;                       РЕГИСТР D = СТАРШИЙ БАЙТ ДЕЛИТЕЛЯ                    ;
;                                                                            ;
;       ВЫХОД:          РЕГИСТР L = МЛАДШИЙ БАЙТ ЧАСТНОГО                    ;
;                       РЕГИСТР H = СТАРШИЙ БАЙТ ЧАСТНОГО                    ;
;                       РЕГИСТР E = МЛАДШИЙ БАЙТ ОСТАТКА                     ;
;                       РЕГИСТР D = СТАРШИЙ БАЙТ ОСТАТКА                     ;
;                                                                            ;
;                       ЕСЛИ НЕТ ОШИБОК, ТО                                  ;
;                         ФЛАГ ПЕРЕНОСА := 0                                 ;
;                       ИНАЧЕ                                                ;
;                         ОШИБКА ДЕЛЕНИЯ НА НУЛЬ                             ;
;                         ФЛАГ ПЕРЕНОСА := 1                                 ;
;                         ЧАСТНОЕ := 0                                       ;
;                         ОСТАТОК := 0                                       ;
;                                                                            ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ                                           ;
;                                                                            ;
;       ВРЕМЯ:          ПРИБЛИЗИТЕЛЬНО ОТ 2630 ДО 3200 ТАКТОВ ДЛЯ 8080       ;
;                       ПРИБЛИЗИТЕЛЬНО ОТ 2480 ДО 2950 ТАКТОВ ДЛЯ 8085       ;
;                                                                            ;
;       РАЗМЕР:         ПРОГРАММА - 127 БАЙТ                                 ;
;                       ДАННЫЕ    -   3 БАЙТА                                ;
;                                                                            ;
;                                                                            ;

        ;ДЕЛЕНИЕ СО ЗНАКОМ
SDIV16:
        ;ОПРЕДЕЛИТЬ ЗНАК ЧАСТНОГО С ПОМОЩЬЮ ОПЕРАЦИИ "ИСКЛЮЧАЮЩЕЕ ИЛИ"
        ; МЕЖДУ СТАРШИМИ БАЙТАМИ ДЕЛИМОГО И ДЕЛИТЕЛЯ. ЧАСТНОЕ
        ; ПОЛОЖИТЕЛЬНОЕ, ЕСЛИ ЗНАКИ СОВПАДАЮТ, И ОТРИЦАТЕЛЬНОЕ В
        ; ПРОТИВНОМ СЛУЧАЕ
        ;
        ;ОСТАТОК ИМЕЕТ ТОТ ЖЕ САМЫЙ ЗНАК, ЧТО И ДЕЛИМОЕ
        MOV     A,H             ;ВЗЯТЬ СТАРШИЙ БАЙТ ДЕЛИМОГО
        STA     SREM            ;СОХРАНИТЬ ЕГО КАК ЗНАК ОСТАТКА
        XRA     D               ;"ИСКЛЮЧАЮЩЕЕ ИЛИ" СО СТАРШИМ БАЙТОМ
                                ; ДЕЛИТЕЛЯ
        STA     SQUOT           ;СОХРАНИТЬ ЗНАК ЧАСТНОГО

        ;ПОЛУЧИТЬ АБСОЛЮТНОЕ ЗНАЧЕНИЕ ДЕЛИТЕЛЯ
        MOV     A,D
        ORA     A
        JP      CHKDE           ;ПЕРЕЙТИ, ЕСЛИ ДЕЛИТЕЛЬ ПОЛОЖИТЕЛЬНЫЙ
        SUB     A               ;ВЫЧЕСТЬ ДЕЛИТЕЛЬ ИЗ НУЛЯ
        SUB     E
        MOV     E,A
        SBB     A               ;ПРОДВИНУТЬ ЗАЕМ (ЕСЛИ БЫЛ ЗАЕМ,ТО
                                ; A = FF)
```

```
        SUB     D
        MOV     D,A

        ;ПОЛУЧИТЬ АБСОЛЮТНОЕ ЗНАЧЕНИЕ ДЕЛИМОГО
CHKDE:
        MOV     A,H
        ORA     A
        JP      DODIV           ;ПЕРЕЙТИ, ЕСЛИ ДЕЛИМОЕ ПОЛОЖИТЕЛЬНОЕ
        SUB     A               ;ВЫЧЕСТЬ ДЕЛИМОЕ ИЗ НУЛЯ
        SUB     L
        MOV     L,A
        SBB     A               ;ПРОДВИНУТЬ ЗАЕМ (ЕСЛИ БЫЛ ЗАЕМ, ТО
                                ; A = FF)
        SUB     H
        MOV     H,A

        ;РАЗДЕЛИТЬ АБСОЛЮТНЫЕ ЗНАЧЕНИЯ
DODIV:  CALL    UDIV16
        RC                      ;ПРИ ДЕЛЕНИИ НА НУЛЬ
                                ; ВЫЙТИ ИЗ ПОДПРОГРАММЫ

        ;СДЕЛАТЬ ЧАСТНОЕ ОТРИЦАТЕЛЬНЫМ, ЕСЛИ ОНО ДОЛЖНО БЫТЬ ТАКОВЫМ
        LDA     SQUOT
        ORA     A
        JP      DOREM           ;ПЕРЕЙТИ, ЕСЛИ ЧАСТНОЕ ПОЛОЖИТЕЛЬНОЕ
        MVI     A,0             ;ВЫЧЕСТЬ ЧАСТНОЕ ИЗ НУЛЯ
        SUB     L
        MOV     L,A
        MVI     A,0
        SBB     H
        MOV     H,A

DOREM:
        ;СДЕЛАТЬ ОСТАТОК ОТРИЦАТЕЛЬНЫМ, ЕСЛИ ОН ДОЛЖЕН БЫТЬ ТАКОВЫМ
        LDA     SREM
        ORA     A
        RP                      ;ВЫЙТИ ИЗ ПОДПРОГРАММЫ, ЕСЛИ ОСТАТОК
                                ; ПОЛОЖИТЕЛЬНЫЙ
        SUB     A               ;ВЫЧЕСТЬ ОСТАТОК ИЗ НУЛЯ
        SUB     E
        MOV     E,A
        SBB     A               ;ПРОДВИНУТЬ ЗАЕМ (ЕСЛИ БЫЛ ЗАЕМ,ТО
                                ; A = FF)
        SUB     D
        MOV     D,A
        RET


        ;ДЕЛЕНИЕ БЕЗ ЗНАКА
UDIV16:
        ;ПРОВЕРИТЬ НА ДЕЛЕНИЕ НА НУЛЬ
        MOV     A,E             ;ПРОВЕРИТЬ ДЕЛИТЕЛЬ
        ORA     D
        JNZ     DIVIDE          ;ПЕРЕЙТИ, ЕСЛИ ДЕЛИТЕЛЬ НЕ РАВЕН НУЛЮ
        LXI     H,0             ;ОШИБКА ДЕЛЕНИЯ НА НУЛЬ
        MOV     D,H
        MOV     E,L
        STC                     ;УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА,
                                ; РЕЗУЛЬТАТ ОШИБОЧНЫЙ
```

```
        RET

DIVIDE: MOV     C,L                     ;BC = ДЕЛИМОЕ/ЧАСТНОЕ
        MOV     B,H
        LXI     H,0                     ;DE = ОСТАТОК
        MVI     A,16                    ;ДЕЛИМОЕ ИМЕЕТ 16 РАЗРЯДОВ
        ORA     A                       ;ДЛЯ НАЧАЛА ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА
DVLOOP:
        STA     COUNT                   ;СОХРАНИТЬ ТЕКУЩЕЕ ЗНАЧЕНИЕ СЧЕТЧИКА

        ;СДВИНУТЬ СЛЕДУЮЩИЙ РАЗРЯД ЧАСТНОГО В РАЗРЯД 0 ДЕЛИМОГО
        ;СДВИНУТЬ СЛЕДУЮЩИЙ СТАРШИЙ РАЗРЯД ДЕЛИМОГО В МЛАДШИЙ РАЗРЯД
        ; ОСТАТКА
        ;BC СОДЕРЖИТ КАК ДЕЛИМОЕ, ТАК И ЧАСТНОЕ. СДВИГАЯ РАЗРЯД ИЗ
        ; СТАРШЕГО БАЙТА ДЕЛИМОГО, МЫ СДВИГАЕМ В РЕГИСТР ИЗ ФЛАГА
        ; ПЕРЕНОСА СЛЕДУЮЩИЙ РАЗРЯД ЧАСТНОГО
        ;HL СОДЕРЖИТ ОСТАТОК
        ;
        ;ВЫПОЛНИТЬ СДВИГ ВЛЕВО 32 РАЗРЯДОВ, СДВИГАЯ ФЛАГ ПЕРЕНОСА В
        ; РЕГИСТР C, РЕГИСТР C В РЕГИСТР B, РЕГИСТР B В РЕГИСТР L,
        ; РЕГИСТР L В РЕГИСТР H
        MOV     A,C
        RAL                             ;ФЛАГ ПЕРЕНОСА (СЛЕДУЮЩИЙ РАЗРЯД
                                        ; ЧАСТНОГО)
        MOV     C,A                     ; В РАЗРЯД 0, РАЗРЯД 7 - ВО ФЛАГ ПЕРЕНОСА
        MOV     A,B
        RAL                             ;СДВИНУТЬ ДРУГОЙ БАЙТ ДЕЛИМОГО
        MOV     B,A                     ; И ЧАСТНОГО
        MOV     A,L
        RAL                             ;СДВИНУТЬ СЛЕДУЮЩИЙ РАЗРЯД ДЕЛИМОГО
        MOV     L,A                     ; В ОСТАТОК
        MOV     A,H
        RAL                             ;СДВИНУТЬ СЛЕДУЮЩИЙ РАЗРЯД ДЕЛИМОГО
        MOV     H,A                     ; В СТАРШИЙ БАЙТ ОСТАТКА

        ;ЕСЛИ ОСТАТОК БОЛЬШЕ ИЛИ РАВЕН ДЕЛИТЕЛЮ, ТО СЛЕДУЮЩИЙ РАЗРЯД
        ; ЧАСТНОГО РАВЕН 1. ЭТОТ РАЗРЯД ПОПАДАЕТ ВО ФЛАГ ПЕРЕНОСА
        PUSH    H                       ;СОХРАНИТЬ ТЕКУЩИЙ ОСТАТОК
        MOV     A,L                     ;ВЫЧЕСТЬ ДЕЛИТЕЛЬ ИЗ ОСТАТКА
        SUB     E
        MOV     L,A
        MOV     A,H
        SBB     D
        MOV     H,A
        CMC                             ;ИНВЕРТИРОВАТЬ ЗАЕМ ДЛЯ ТОГО, ЧТОБЫ 1
                                        ; ВО ФЛАГЕ ПЕРЕНОСА УКАЗЫВАЛА, ЧТО
                                        ; ВЫЧИТАНИЕ ПРОШЛО УСПЕШНО
                                        ; (ЭТО СЛЕДУЮЩИЙ РАЗРЯД ЧАСТНОГО)
        JC      DROP                    ;ПЕРЕЙТИ, ЕСЛИ ОСТАТОК >= ДЕЛИМОМУ
        XTHL                            ;В ПРОТИВНОМ СЛУЧАЕ ВОССТАНОВИТЬ ОСТАТОК

DROP:
        INX     SP                      ;ОПУСТИТЬ ОСТАТОК ИЗ ВЕРШИНЫ СТЕКА
        INX     SP
        LDA     COUNT
        DCR     A
```

```
        JNZ     DVLOOP          ;ПРОДОЛЖИТЬ ТО ЖЕ САМОЕ ДЛЯ ВСЕХ РАЗРЯДОВ

        ;СДВИНУТЬ ПОСЛЕДНИЙ ПЕРЕНОС В ЧАСТНОЕ
        XCHG                    ;DE = ОСТАТОК
        MOV     A,C
        RAL                     ;ПОСЛЕДНИЙ РАЗРЯД ЧАСТНОГО В РАЗРЯД 0
        MOV     L,A             ;ЧАСТНОЕ В HL
        MOV     A,B
        RAL
        MOV     H,A
        ORA     A               ;РЕЗУЛЬТАТ ПРАВИЛЬНЫЙ,
                                ; ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА
        RET

        ;ДАННЫЕ
SQUOT:  DS      1               ;ЗНАК ЧАСТНОГО
SREM:   DS      1               ;ЗНАК ОСТАТКА
COUNT:  DS      1               ;СЧЕТЧИК ЦИКЛА ДЕЛЕНИЯ


;                                                                       ;
;                                                                       ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                               ;
;                                                                       ;
;                                                                       ;

SC6C:
        ;ДЕЛЕНИЕ СО ЗНАКОМ, -1023 / 123
        LXI     H,-1023         ;HL = ДЕЛИМОЕ
        LXI     D,123           ;DE = ДЕЛИТЕЛЬ
        CALL    SDIV16          ;ЧАСТНОЕ ОТ -1023 / 123 = -8
                                ; L = F8H
                                ; H = FFH
                                ;ОСТАТОК ОТ -1023 / 123 = -39
                                ; E = D9H
                                ; D = FFH

        ;ДЕЛЕНИЕ БЕЗ ЗНАКА, 64513 / 123
        LXI     H,64513
        LXI     D,123
        CALL    UDIV16          ;ЧАСТНОЕ ОТ 64513 / 123 = 524
                                ; L = 0CH
                                ; H = 02H
                                ;ОСТАТОК ОТ 64513 / 123 = 61
                                ; E = 3DH
                                ; D = 00H

        JMP     SC6C

        END
```

6D. ШЕСТНАДЦАТИРАЗРЯДНОЕ СРАВНЕНИЕ (CMP16)

Сравниваются два 16-разрядных операнда и соответствующим образом устанавливаются флаги. Флаг нуля всегда указывает, были ли операнды равны. Если операнды были беззнаковые, то флаг переноса указывает, какой из них больше (флаг переноса = 1, если вычитаемое больше. и 0 — в противном

случае). Если операнды имеют знаки, то флаг знака указывает, какой из них больше (флаг знака равен 1, если вычитаемое больше, и 0 – в противном случае); при этом учитывается переполнение по дополнению до двух, и если оно происходит, то флаг знака инвертируется.

Процедура. Сначала проверяется, может ли произойти переполнение по дополнению до двух. Это возможно только в том случае, если знаки операндов различаются. Если переполнение по дополнению до двух возможно, то вычитается младший байт вычитаемого из уменьшаемого. Если младшие байты равны, то устанавливаются флаги по результату вычитания старших байтов. Если младшие байты не равны, перед выходом должен очиститься флаг нуля (с помощью логической операции ИЛИ с 1, но при неизменных остальных флагах). Если переполнение по дополнению до двух может произойти, то устанавливается флаг знака по знаку уменьшаемого. Это выполняется с помощью загрузки в аккумулятор старшего байта уменьшаемого перед установкой флага знака.

Используемые регистры: AF, DE, HL.
Время выполнения: приблизительно от 57 до 81 такта (8080) или от 51 до 69 тактов (8085).
Размер программы: 36 байт.
Память, необходимая для данных: отсутствует.

УСЛОВИЯ НА ВХОДЕ

Уменьшаемое в регистрах H и L.
Вычитаемое в регистрах D и E.

УСЛОВИЯ НА ВЫХОДЕ

Флаги устанавливаются так же, как при вычитании вычитаемого из уменьшаемого, с поправкой, если происходит переполнение по дополнению до двух.
Флаг нуля = 1, если вычитаемое и уменьшаемое равны; 0 – если не равны.
Флаг переноса = 1, если вычитаемое больше уменьшаемого для чисел без знаков; 0, если вычитаемое меньше или равно уменьшаемому.
Флаг знака = 1, если уменьшаемое больше вычитаемого для чисел со знаками; 0, если вычитаемое меньше или равно уменьшаемому. Этот флаг корректируется (инвертируется), если происходит переполнение по дополнению до двух.

ПРИМЕРЫ

1. Данные: уменьшаемое (HL) = $03E1_{16}$,
вычитаемое (DE) = $07E4_{16}$.
Результат: флаг переноса = 1, т. е. вычитаемое больше (для чисел без знаков),
флаг нуля = 0, т. е. операнды не равны,
флаг знака = 1, т. е. вычитаемое больше (для чисел со знаками).

2. Данные: уменьшаемое (HL) = $C51A_{16}$,
вычитаемое (DE) = $C51A_{16}$.
Результат: флаг переноса = 0, т. е. вычитаемое не больше (для чисел без знака),
флаг нуля = 1, т. е. операнды равны,
флаг знака = 0, т. е. вычитаемое не больше (для чисел со знаком).

3. Данные: уменьшаемое (HL) = $A45D_{16}$,
вычитаемое (DE) = $77E1_{16}$.

Результат: флаг переноса = 0, т. е. вычитаемое не больше (для чисел без знака),
флаг нуля = 0, т. е. операнды не равны,
флаг знака = 1, т. е. вычитаемое больше (для чисел со знаками).

```
;                                                                     ;
;                                                                     ;
;                                                                     ;
;                                                                     ;
;       ЗАГОЛОВОК:      16-РАЗРЯДНОЕ СРАВНЕНИЕ                        ;
;       ИМЯ:            CMP16                                         ;
;                                                                     ;
;                                                                     ;


;                                                                     ;
;       НАЗНАЧЕНИЕ:     СРАВНИВАЕТ ДВА 16-РАЗРЯДНЫХ СЛОВА СО ЗНАКОМ ИЛИ ;
;                       БЕЗ ЗНАКА И ВОЗВРАЩАЕТ ФЛАГИ ПЕРЕНОСА (C),    ;
;                       НУЛЯ (Z) И ЗНАКА (S) РАВНЫМИ 1 ИЛИ 0          ;
;                                                                     ;
;       ВХОД:           РЕГИСТР L = МЛАДШИЙ БАЙТ УМЕНЬШАЕМОГО         ;
;                       РЕГИСТР H = СТАРШИЙ БАЙТ УМЕНЬШАЕМОГО         ;
;                       РЕГИСТР E = МЛАДШИЙ БАЙТ ВЫЧИТАЕМОГО          ;
;                       РЕГИСТР D = СТАРШИЙ БАЙТ ВЫЧИТАЕМОГО          ;
;                                                                     ;
;       ВЫХОД:          ВОЗВРАЩАЮТСЯ ФЛАГИ, КАК РЕЗУЛЬТАТ ОПЕРАЦИИ    ;
;                         УМЕНЬШАЕМОЕ - ВЫЧИТАЕМОЕ                    ;
;                       ЕСЛИ КАК УМЕНЬШАЕМОЕ, ТАК И ВЫЧИТАЕМОЕ ЯВЛЯЮТСЯ ;
;                         ДОПОЛНЕНИЯМИ ДО ДВУХ, ТО ИСПОЛЬЗУЮТСЯ ФЛАГИ ;
;                         Z И S;                                      ;
;                       ИНАЧЕ ИСПОЛЬЗУЮТСЯ ФЛАГИ Z И C.               ;
;                        ЕСЛИ УМЕНЬШАЕМОЕ = ВЫЧИТАЕМОМУ, ТО           ;
;                           Z=1,S=0,C=0                               ;
;                        ЕСЛИ УМЕНЬШАЕМОЕ > ВЫЧИТАЕМОГО, ТО           ;
;                           Z=0,S=0,C=0                               ;
;                        ЕСЛИ УМЕНЬШАЕМОЕ < ВЫЧИТАЕМОГО, ТО           ;
;                           Z=0,S=1,C=1                               ;
;                                                                     ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF,DE,HL                               ;
;                                                                     ;
;       ВРЕМЯ:          ПРИБЛИЗИТЕЛЬНО ОТ 57 ДО 81 ТАКТОВ ДЛЯ 8080    ;
;                       ПРИБЛИЗИТЕЛЬНО ОТ 51 ДО 69 ТАКТОВ ДЛЯ 8085    ;
;                                                                     ;
;       РАЗМЕР:         ПРОГРАММА - 36 БАЙТ                           ;
;                                                                     ;
;                                                                     ;

        ;ПЕРЕПОЛНЕНИЕ ЧИСЕЛ, ЯВЛЯЮЩИХСЯ ДОПОЛНЕНИЯМИ ДО ДВУХ, ВОЗМОЖНО
        ; ТОЛЬКО В СЛУЧАЕ, ЕСЛИ ЗНАКИ ОПЕРАНДОВ РАЗЛИЧАЮТСЯ

CMP16:
        MOV     A,D
        XRA     H               ;ЗНАКИ ОПЕРАНДОВ РАЗЛИЧАЮТСЯ?
        JM      DIFF            ;ПЕРЕЙТИ, ЕСЛИ ЗНАКИ РАЗЛИЧАЮТСЯ

        ;ПЕРЕПОЛНЕНИЕ НЕВОЗМОЖНО - ВЫПОЛНИТЬ СРАВНЕНИЕ БЕЗ ЗНАКА
        MOV     A,L             ;СРАВНИТЬ МЛАДШИЕ БАЙТЫ
```

```
        SUB     E
        JZ      EQUAL                ;ПЕРЕЙТИ, ЕСЛИ МЛАДШИЕ БАЙТЫ РАВНЫ

        ;МЛАДШИЕ БАЙТЫ НЕ РАВНЫ, СРАВНИТЬ СТАРШИЕ БАЙТЫ
        ;ЗАПОМНИМ, ЧТО ФЛАГ НУЛЯ ПОЗДНЕЕ ДОЛЖЕН БЫТЬ ОЧИЩЕН
        MOV     A,H                  ;СРАВНИТЬ СТАРШИЕ БАЙТЫ
        SBB     D
        JC      CYSET                ;ВЫЙТИ С ФЛАГОМ ПЕРЕНОСА = 1,
                                     ; ФЛАГ НУЛЯ = 0
        JNC     CYCLR                ;ВЫЙТИ С ФЛАГОМ ПЕРЕНОСА = 0,
                                     ; ФЛАГ НУЛЯ = 0

        ;МЛАДШИЕ БАЙТЫ РАВНЫ, УСТАНОВИТЬ ФЛАГИ, СРАВНИВАЯ СТАРШИЕ
        ; БАЙТЫ С УЧЕТОМ ЗАЕМА
EQUAL:
        MOV     A,H
        SBB     D
        RET

        ;ЗНАКИ ОПЕРАНДОВ РАЗЛИЧАЮТСЯ, СРАВНИТЬ, УСТАНОВИВ ФЛАГ ЗНАКА
        ; РАВНЫМ ЗНАКУ УМЕНЬШАЕМОГО
DIFF:
        MOV     A,L
        SUB     E                    ;ВЫЧЕСТЬ МЛАДШИЕ БАЙТЫ
        MOV     A,H
        SBB     D                    ;ВЫЧЕСТЬ СТАРШИЕ БАЙТЫ
        MOV     A,H                  ;ЗАГРУЗИТЬ СТАРШИЙ БАЙТ ВЫЧИТАЕМОГО,
                                     ; ЧТОБЫ ПОЛУЧИТЬ ПРАВИЛЬНЫЙ ЗНАК
        JNC     CYCLR

        ;ВЫЙТИ С ФЛАГОМ ПЕРЕНОСА = 1 И ФЛАГОМ НУЛЯ = 0
        ;ПРОЛЕМА ЗДЕСЬ СОСТОИТ В ТОМ, ЧТО ORI ВСЕГДА ОЧИЩАЕТ
        ; ФЛАГ ПЕРЕНОСА
CYSET:
        ORI     1                    ;ФЛАГ НУЛЯ = 0, ФЛАГ ЗНАКА = РАЗРЯД 7
        STC                          ;ФЛАГ ПЕРЕНОСА = 1
        RET

        ;ВЫХОД С ФЛАГОМ ПЕРЕНОСА = 0 И ФЛАГОМ НУЛЯ = 0
CYCLR:
        ORI     1                    ;ФЛАГ НУЛЯ = 0, ФЛАГ ЗНАКА = РАЗРЯД 7
                                     ; И ФЛАГ ПЕРЕНОСА = 0
        RET


;                                                                      ;
;                                                                      ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                              ;
;                                                                      ;
;                                                                      ;

SC6D:
        ;СРАВНИТЬ -32768 (ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО 8000) И 1
        ;ТАК КАК -32768 - НАИМЕНЬШЕЕ ОТРИЦАТЕЛЬНОЕ 16-РАЗРЯДНОЕ ЧИСЛО,
        ; ТО ЭТО СРАВНЕНИЕ БЕЗУСЛОВНО ВЫЗОВЕТ ПЕРЕПОЛНЕНИЕ
        LXI     H,-32768
        LXI     D,1
        CALL    CMP16                ;CY = 0, Z = 0, S = 1
```

```
;СРАВНИТЬ -4 (ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО FFFC)
;  И -1 (ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО FFFF)
LXI     H,-4
LXI     D,-1
CALL    CMP16           ;CY = 1, Z = 0, S = 1

;СРАВНИТЬ -1234 И -1234
LXI     H,-1234
LXI     D,-1234
CALL    CMP16           ;CY = 0, Z = 1, S = 0

JMP     SC6D

END
```

6E. ДВОИЧНОЕ СЛОЖЕНИЕ ЧИСЕЛ С ПОВЫШЕННОЙ ТОЧНОСТЬЮ (MPBADD)

Складываются два многобайтных двоичных числа. Числа хранятся в памяти таким образом, что их самые младшие по значению байты занимают самые младшие адреса. Сумма заменяет первое слагаемое. Длина чисел равна 255 байт или меньше.

Процедура. Сначала очищается флаг переноса, а затем складываются операнды, по байту за раз, начиная с самых младших по значению байтов. В конце флаг переноса отражает результат сложения самых старших байтов. Длина 0 вызывает немедленный выход без сложения.

Используемые регистры: AF, B, DE, HL.
Время выполнения: 46 тактов на байт плюс 24 такта (8080) или 47 тактов на байт плюс 21 такт (8085).
Размер программы: 13 байт.
Память, необходимая для данных: отсутствует.
Специальный случай: длина 0 вызывает немедленный выход, при этом первое слагаемое остается без изменения. Флаг переноса очищается.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес первого слагаемого в регистрах H и L.
Базовый адрес второго слагаемого в регистрах D и E.
Длина операндов в байтах в регистре B.

УСЛОВИЯ НА ВЫХОДЕ

Первое слагаемое замещается суммой первого и второго слагаемых.

ПРИМЕР

1. Данные: длина операндов (в байтах) = 6,
первое слагаемое = $19D028A193EA_{16}$,
второе слагаемое = $293EABF059C7_{16}$.
Результат: первое слагаемое = $430ED491EDB1_{16}$,
флаг переноса = 0.

```
;                                                                          ;
;                                                                          ;
;                                                                          ;
;                                                                          ;
;       ЗАГОЛОВОК:      ДВОИЧНОЕ СЛОЖЕНИЕ ЧИСЕЛ С ПОВЫШЕННОЙ ТОЧНОСТЬЮ     ;
;       ИМЯ:            MPBADD                                             ;
;                                                                          ;
;                                                                          ;


;                                                                          ;
;       НАЗНАЧЕНИЕ:     СКЛАДЫВАЕТ ДВА МАССИВА БАЙТОВ, СОДЕРЖАЩИХ          ;
;                       ДВОИЧНЫЕ ЧИСЛА                                     ;
;                       МАССИВ1 := МАССИВ1 + МАССИВ2                       ;
;                                                                          ;
;       ВХОД:           РЕГИСТРЫ H И L = БАЗОВЫЙ АДРЕС МАССИВА 1           ;
;                       РЕГИСТРЫ D И E = БАЗОВЫЙ АДРЕС МАССИВА 2           ;
;                       РЕГИСТР B = ДЛИНА МАССИВОВ                         ;
;                                                                          ;
;                         КАЖДЫЙ МАССИВ ДОЛЖЕН СОДЕРЖАТЬ ОДНО ДВОИЧНОЕ     ;
;                         ЧИСЛО БЕЗ ЗНАКА С МАКСИМАЛЬНОЙ ДЛИНОЙ 255        ;
;                         БАЙТ. ARRAY[0] ЯВЛЯЕТСЯ МЛАДШИМ БАЙТОМ, А        ;
;                         ARRAY[LENGTH-1] - СТАРШИМ БАЙТОМ; ЗДЕСЬ          ;
;                         ARRAY - БАЗОВЫЙ АДРЕС МАССИВА, А LENGTH -        ;
;                         ДЛИНА.                                           ;
;                                                                          ;
;       ВЫХОД:          МАССИВ1 := МАССИВ1 + МАССИВ2                       ;
;                                                                          ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF,B,DE,HL                                  ;
;                                                                          ;
;       ВРЕМЯ:          46 ТАКТОВ НА БАЙТ ПЛЮС 24 ТАКТА ДЛЯ 8080           ;
;                       47 ТАКТОВ НА БАЙТ ПЛЮС 21 ТАКТ  ДЛЯ 8085           ;
;                                                                          ;
;       РАЗМЕР:         ПРОГРАММА - 13 БАЙТ                                ;
;                                                                          ;
;                                                                          ;

MPBADD:
        ;ДЛЯ НАЧАЛА ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА, ВЫЙТИ, ЕСЛИ ДЛИНА МАССИВА 0
        MOV     A,B
        ANA     A               ;ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА, ПРОВЕРИТЬ ДЛИНУ
        RZ                      ;ВОЗВРАТИТЬСЯ, ЕСЛИ ДЛИНА = НУЛЮ
LOOP:
        LDAX    D               ;ВЗЯТЬ СЛЕДУЮЩИЙ БАЙТ
        ADC     M               ;СЛОЖИТЬ БАЙТЫ
        MOV     M,A             ;ЗАПОМНИТЬ СУММУ
        INX     H               ;УВЕЛИЧИТЬ УКАЗАТЕЛЬ МАССИВА1
        INX     D               ;УВЕЛИЧИТЬ УКАЗАТЕЛЬ МАССИВА2
        DCR     B
        JNZ     LOOP            ;ПРОДОЛЖАТЬ, ПОКА СЧЕТЧИК НЕ БУДЕТ = 0
        RET
;                                                                          ;
;                                                                          ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                                  ;
;                                                                          ;
;                                                                          ;
```

```
SC6E:
        LXI     H,AY1           ;HL = БАЗОВЫЙ АДРЕС МАССИВА 1
        LXI     D,AY2           ;DE = БАЗОВЫЙ АДРЕС МАССИВА 2
        MVI     B,SZAYS         ;B = ДЛИНА МАССИВОВ В БАЙТАХ
        CALL    MPBADD          ;СЛОЖИТЬ МАССИВЫ
                                ;        AY1+0 = 56H
                                ;        AY1+1 = 13H
                                ;        AY1+2 = CFH
                                ;        AY1+3 = 8AH
                                ;        AY1+4 = 67H
                                ;        AY1+5 = 45H
                                ;        AY1+6 = 23H
                                ;        AY1+7 = 01H

        JMP     SC6E

SZAYS   EQU     8               ;ДЛИНА МАССИВОВ В БАЙТАХ
AY1:
        DB      0EFH
        DB      0CDH
        DB      0ABH
        DB      089H
        DB      067H
        DB      045H
        DB      023H
        DB      001H

AY2:
        DB      067H
        DB      045H
        DB      023H
        DB      001H
        DB      0
        DB      0
        DB      0
        DB      0

        END
```

6F. ДВОИЧНОЕ ВЫЧИТАНИЕ ЧИСЕЛ С ПОВЫШЕННОЙ ТОЧНОСТЬЮ (MPBSUB)

Вычитаются два многобайтных беззнаковых двоичных числа. Оба числа хранятся в памяти таким образом, что их самые младшие по значению байты занимают самые младшие адреса. Разность замещает уменьшаемое. Длина чисел равна 255 байт или меньше.

Процедура. Сначала очищается флаг переноса, а затем вычисляются операнды, по байту за раз, начиная с самых младших по значению байтов. В конце флаг переноса отражает результат вычитания самых старших байтов. Длина 0 вызывает немедленный выход без вычитания.

Используемые регистры: AF, B, DE, HL.
Время выполнения: 46 тактов на байт плюс 28 тактов (8080) или 47 тактов на байт плюс 25 тактов (8085).
Размер программы: 14 байт.

Память, необходимая для данных: отсутствует.
Специальный случай: длина 0 вызывает немедленный выход без изменения уменьшаемого (т. е. разность равна уменьшаемому).
Флаг переноса очищается.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес уменьшаемого в регистрах H и L.
Базовый адрес вычитаемого в регистрах D и E.
Длина операндов в байтах в регистре B.

УСЛОВИЯ НА ВЫХОДЕ

Уменьшаемое заменяется разностью уменьшаемого и вычитаемого.

ПРИМЕР

1. Данные: длина операндов (в байтах) = 4,
уменьшаемое = $2F5BA7C3_{16}$,
вычитаемое = $14DF35B8_{16}$.
Результат: уменьшаемое $1A7C720B_{16}$,
флаг переноса равен 0, так как нет необходимости в заеме.

```
;                                                                        ;
;                                                                        ;
;                                                                        ;
;                                                                        ;
;       ЗАГОЛОВОК:      ДВОИЧНОЕ ВЫЧИТАНИЕ ЧИСЕЛ С ПОВЫШЕННОЙ ТОЧНОСТЬЮ ;
;       ИМЯ:            MPBSUB                                          ;
;                                                                        ;
;                                                                        ;


;                                                                        ;
;       НАЗНАЧЕНИЕ:     ВЫЧИТАЕТ ДВА МАССИВА БАЙТОВ, СОДЕРЖАЩИХ         ;
;                       ДВОИЧНЫЕ ЧИСЛА                                   ;
;                       УМЕНЬШАЕМОЕ := УМЕНЬШАЕМОЕ - ВЫЧИТАЕМОЕ          ;
;                                                                        ;
;       ВХОД:           РЕГИСТРЫ H И L = БАЗОВЫЙ АДРЕС УМЕНЬШАЕМОГО      ;
;                       РЕГИСТРЫ D И E = БАЗОВЫЙ АДРЕС ВЫЧИТАЕМОГО       ;
;                       РЕГИСТР B = ДЛИНА МАССИВОВ                       ;
;                                                                        ;
;                         КАЖДЫЙ МАССИВ ДОЛЖЕН СОДЕРЖАТЬ ОДНО ДВОИЧНОЕ   ;
;                         ЧИСЛО БЕЗ ЗНАКА С МАКСИМАЛЬНОЙ ДЛИНОЙ 255      ;
;                         БАЙТ. ARRAY[0] ЯВЛЯЕТСЯ МЛАДШИМ БАЙТОМ, А      ;
;                         ARRAY[LENGTH-1] - СТАРШИМ БАЙТОМ; ЗДЕСЬ        ;
;                         ARRAY - БАЗОВЫЙ АДРЕС МАССИВА, А LENGTH -      ;
;                         ДЛИНА.                                         ;
;                                                                        ;
;       ВЫХОД:          УМЕНЬШАЕМОЕ := УМЕНЬШАЕМОЕ - ВЫЧИТАЕМОЕ          ;
;                                                                        ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF,B,DE,HL                                ;
;                                                                        ;
;       ВРЕМЯ:          46 ТАКТОВ НА БАЙТ ПЛЮС 28 ТАКТОВ ДЛЯ 8080        ;
;                       47 ТАКТОВ НА БАЙТ ПЛЮС 25 ТАКТОВ ДЛЯ 8085        ;
;                                                                        ;
;       РАЗМЕР:         ПРОГРАММА - 14 БАЙТ                              ;
```

```
;                                                                          ;
;                                                                          ;

MPBSUB:
        ;ДЛЯ НАЧАЛА ОЧИСТИТЬ ЗАЕМ, ВЫЙТИ, ЕСЛИ ДЛИНА МАССИВОВ РАВНА 0
        MOV     A,B
        ANA     A               ;ОЧИСТИТЬ ЗАЕМ, ПРОВЕРИТЬ ДЛИНУ
        RZ                      ;ВОЗВРАТИТЬСЯ, ЕСЛИ ДЛИНА = НУЛЮ
        XCHG                    ;ПОМЕНЯТЬ МЕСТАМИ УКАЗАТЕЛИ МАССИВОВ,
                                ; ТАК ЧТОБЫ ПАРА РЕГИСТРОВ HL УКАЗЫВАЛА
                                ; НА ВЫЧИТАЕМОЕ
LOOP:
        LDAX    D               ;ВЗЯТЬ СЛЕДУЮЩИЙ БАЙТ УМЕНЬШАЕМОГО
        SBB     M               ;ВЫЧЕСТЬ БАЙТЫ
        STAX    D               ;ЗАПОМНИТЬ РАЗНОСТЬ
        INX     D               ;УВЕЛИЧИТЬ УКАЗАТЕЛЬ УМЕНЬШАЕМОГО
        INX     H               ;УВЕЛИЧИТЬ УКАЗАТЕЛЬ ВЫЧИТАЕМОГО
        DCR     B
        JNZ     LOOP            ;ПРОДОЛЖАТЬ, ПОКА СЧЕТЧИК НЕ БУДЕТ
                                ; РАВЕН 0
        RET


;                                                                          ;
;                                                                          ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                                  ;
;                                                                          ;
;                                                                          ;

SC6F:
        LXI     H,AY1           ;HL = БАЗОВЫЙ АДРЕС УМЕНЬШАЕМОГО
        LXI     D,AY2           ;DE = БАЗОВЫЙ АДРЕС ВЫЧИТАЕМОГО
        MVI     B,SZAYS         ;B = ДЛИНА МАССИВОВ В БАЙТАХ
        CALL    MPBSUB          ;ВЫЧЕСТЬ МАССИВЫ
                                ;          AY1+0 = 88H
                                ;          AY1+1 = 88H
                                ;          AY1+2 = 88H
                                ;          AY1+3 = 88H
                                ;          AY1+4 = 67H
                                ;          AY1+5 = 45H
                                ;          AY1+6 = 23H
                                ;          AY1+7 = 01H

        JMP     SC6F

SZAYS   EQU     8               ;ДЛИНА МАССИВОВ В БАЙТАХ
AY1:
        DB      0EFH
        DB      0CDH
        DB      0ABH
        DB      089H
        DB      067H
        DB      045H
        DB      023H
        DB      001H
```

```
AY2:
        DB      067H
        DB      045H
        DB      023H
        DB      001H
        DB      0
        DB      0
        DB      0
        DB      0

        END
```

6G. ДВОИЧНОЕ УМНОЖЕНИЕ ЧИСЕЛ С ПОВЫШЕННОЙ ТОЧНОСТЬЮ (MPBMUL)

Умножаются два многобайтных беззнаковых двоичных числа. Оба числа хранятся в памяти таким образом, что их самые младшие по значению байты занимают самые младшие адреса. Произведение замещает множимое. Длина операндов равна 255 байт или меньше. Чтобы сохранялась совместимость с другими двоичными операциями повышенной точности, возвращаются только младшие по значению байты произведения.

Процедура. Используется обычный алгоритм сдвигов и сложения, при этом множитель добавляется к промежуточному произведению каждый раз, когда в множимом находится единичный разряд. Промежуточное произведение и множимое сдвигаются на число разрядов в множимом плюс один; этот дополнительный цикл сдвигает окончательный флаг переноса в произведение. Полное беззнаковое промежуточное произведение двойной длины хранится в ячейках памяти, начиная с HIPROD (старшие по значению байты), и в множимом (младшие по значению байты). Младшие байты произведения замещают множимое по мере того, как оно сдвигается и проверяется на единичные разряды. Длина 0 вызывает выход без умножения.

Используемые регистры: все.
Время выполнения: зависит от длины операндов и числа единичных разрядов в множимом (требующих действительного сложения). Если среднее число единичных разрядов в множимом составляет 4 на 1 байт, то время выполнения приблизительно будет равно $792 * LENGTH^2 + 924 * LENGTH + 300$, где LENGTH – число байтов в операндах.
Размер программы: 116 байт.
Память, необходимая для данных: 261 байт в любом месте ОЗУ. Это область для временного хранения старших по значению байтов произведения (255 байт, начиная с адреса HIPROD), счетчика цикла (2 байта, начиная с адреса COUNT), адреса байта, следующего непосредственно за старшим по значению байтом старшей части произведения (2 байта, начиная с адреса ENDHP), и базового адреса множителя (2 байта, начиная с адреса MLIER).
Специальный случай: длина 0 вызывает немедленный выход с произведением, равным множимому. Флаг переноса очищается.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес множимого в регистрах H и L.
Базовый адрес множителя в регистрах D и E.
Длина операндов в байтах в регистре B.

УСЛОВИЯ НА ВЫХОДЕ

Множимое замещается множимым, умноженным на множитель.

ПРИМЕР

1. Данные: длина операндов (в байтах) = 04,
   множимое = $0005D1F7_{16}$,
   множитель = $00000AB1_{16}$.
   Результат: множимое = $3E39D1C7_{16}$.

Заметим, что для совместимости с другими арифметическими операциями повышенной точности MPBMUL возвращает только младшие байты произведения (т. е. те байты, которые имеют множимое и множитель). Старшие по значению разряды произведения доступны, начиная с адреса их младшего байта HIPROD. У пользователя может появиться потребность в проверке этих байтов на возможность переполнения, или для расширения операндов с помощью дополнительных нулей.

```
;                                                                   ;
;                                                                   ;
;                                                                   ;
;                                                                   ;
;       ЗАГОЛОВОК:      ДВОИЧНОЕ УМНОЖЕНИЕ ЧИСЕЛ С ПОВЫШЕННОЙ ТОЧНОСТЬЮ ;
;       ИМЯ:            MPBMUL                                      ;
;                                                                   ;
;                                                                   ;

;                                                                   ;
;       НАЗНАЧЕНИЕ:     УМНОЖАЕТ ДВА МАССИВА БАЙТ, СОДЕРЖАЩИХ       ;
;                        ДВОИЧНЫЕ ЧИСЛА                             ;
;                       МНОЖИМОЕ := МНОЖИМОЕ * МНОЖИТЕЛЬ            ;
;                                                                   ;
;       ВХОД:           РЕГИСТРЫ H И L = БАЗОВЫЙ АДРЕС МНОЖИМОГО    ;
;                       РЕГИСТРЫ D И E = БАЗОВЫЙ АДРЕС МНОЖИТЕЛЯ    ;
;                       РЕГИСТР B = ДЛИНА МАССИВОВ В БАЙТАХ         ;
;                                                                   ;
;                         КАЖДЫЙ МАССИВ ДОЛЖЕН СОДЕРЖАТЬ ОДНО ДВОИЧНОЕ ;
;                         ЧИСЛО БЕЗ ЗНАКА  С МАКСИМАЛЬНОЙ ДЛИНОЙ 255 ;
;                         БАЙТ. ARRAY[0] ЯВЛЯЕТСЯ МЛАДШИМ БАЙТОМ, А ;
;                         ARRAY[LENGTH-1] - СТАРШИМ БАЙТОМ; ЗДЕСЬ   ;
;                         ARRAY - БАЗОВЫЙ АДРЕС МАССИВА, А LENGTH - ;
;                         ДЛИНА.                                    ;
;                                                                   ;
;       ВЫХОД:          МНОЖИМОЕ := МНОЖИМОЕ * МНОЖИТЕЛЬ            ;
;                                                                   ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ                                  ;
;                                                                   ;
;       ВРЕМЯ:          ЕСЛИ СЧИТАТЬ, ЧТО СРЕДНЕЕ ЧИСЛО РАЗРЯДОВ,   ;
;                       УСТАНОВЛЕННЫХ В 1, В МНОЖИМОМ РАВНО 4 * ДЛИНА, ;
;                       ТО ВРЕМЯ ПРИБЛИЗИТЕЛЬНО РАВНО               ;
;                        (792 * ДЛИНА^2) + (924 * ДЛИНА) + 300 ТАКТОВ ;
;                                                                   ;
;       РАЗМЕР:         ПРОГРАММА - 113 БАЙТ                        ;
;                       ДАННЫЕ    - 261 БАЙТ                        ;
;                                                                   ;
;                                                                   ;
```

```
MPBMUL:
        ;ВЫЙТИ, ЕСЛИ ДЛИНА МАССИВОВ РАВНА НУЛЮ
        MOV     A,B
        ANA     A               ;ДЛИНА МАССИВОВ = 0?
        RZ                      ;ДА, ВЫЙТИ

        ;ПОЛУЧИТЬ УКАЗАТЕЛИ КОНЦОВ МАССИВОВ
        MOV     C,B
        MVI     B,0             ;BC = ДЛИНА
        DAD     B               ;КОНЕЦ = БАЗА + ДЛИНА
        XCHG                    ;DE УКАЗЫВАЕТ НА КОНЕЦ МНОЖИМОГО
        SHLD    MLIER           ;СОХРАНИТЬ АДРЕС МНОЖИТЕЛЯ
        LXI     H,HIPROD
        DAD     B
        SHLD    ENDHP           ;СОХРАНИТЬ АДРЕС В КОНЦЕ МАССИВА HIPROD

        ;УСТАНОВИТЬ СЧЕТЧИК РАВНЫМ ЧИСЛУ РАЗРЯДОВ В МАССИВЕ ПЛЮС 1
        ; СЧЕТЧИК := (ДЛИНА * 8) + 1
        MOV     L,C             ;ПЕРЕСЛАТЬ ДЛИНУ В HL
        MOV     H,B
        DAD     H               ;ДЛИНА * 8, СДВИНУТЬ 3 РАЗА ВЛЕВО
        DAD     H
        DAD     H
        INX     H               ;ПРИБАВИТЬ 1
        SHLD    COUNT           ;СОХРАНИТЬ ЧИСЛО РАЗРЯДОВ,
                                ; КОТОРЫЕ НЕОБХОДИМО ОБРАБОТАТЬ

        ;ЗАПОЛНИТЬ МАССИВ СТАРШЕЙ ЧАСТИ ПРОИЗВЕДЕНИЯ НУЛЯМИ
ZEROPD:
        MOV     B,C             ;B = ДЛИНА В БАЙТАХ
        LXI     H,HIPROD        ;ВЗЯТЬ АДРЕС МАССИВА HIPROD
ZEROLP:
        MVI     M,0             ;ОЧИСТИТЬ БАЙТ МАССИВА HIPROD
        INX     H
        DCR     B
        JNZ     ZEROLP          ;ПРОДОЛЖАТЬ, ПОКА МАССИВ HIPROD
                                ; НЕ БУДЕТ ЗАПОЛНЕН НУЛЯМИ

        ;УМНОЖИТЬ, ИСПОЛЬЗУЯ АЛГОРИТМ СДВИГА И СЛОЖЕНИЯ
        ANA     A               ;ДЛЯ ПЕРВОГО РАЗА ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА
LOOP:
        ;СДВИНУТЬ ФЛАГ ПЕРЕНОСА В МАССИВ HIPROD, А МЛАДШИЙ РАЗРЯД
        ; МАССИВА HIPROD - ВО ФЛАГ ПЕРЕНОСА
        MOV     B,C             ;ВЗЯТЬ ДЛИНУ
        LHLD    ENDHP           ;ВЗЯТЬ АДРЕС ПОСЛЕДНЕГО БАЙТА МАССИВА
                                ; HIPROD + 1
SRPLP:
        DCX     H               ;ПЕРЕЙТИ НАЗАД К СЛЕДУЮЩЕМУ БАЙТУ
        MOV     A,M
        RAR                     ;СДВИНУТЬ ЦИКЛИЧЕСКИ БАЙТ МАССИВА HIPROD
        MOV     M,A             ;ЗАПОМНИТЬ ЕГО
        DCR     B
        JNZ     SRPLP           ;ПРОДОЛЖАТЬ, ПОКА ИНДЕКС НЕ БУДЕТ
                                ; РАВЕН 0

        ;СДВИНУТЬ ФЛАГ ПЕРЕНОСА (СЛЕДУЮЩИЙ МЛАДШИЙ РАЗРЯД МАССИВА
        ; HIPROD) В СТАРШИЙ РАЗРЯД МНОЖИМОГО. ПРИ ЭТОМ ТАКЖЕ
```

```
        ; СЛЕДУЮЩИЙ РАЗРЯД МНОЖИМОГО СДВИГАЕТСЯ ВО ФЛАГ ПЕРЕНОСА
        MOV     L,E                 ;HL = АДРЕС КОНЦА МНОЖИМОГО
        MOV     H,D
        MOV     B,C                 ;B = ДЛИНА В БАЙТАХ
SRA1LP:
        DCX     H                   ;ПЕРЕЙТИ НАЗАД К СЛЕДУЮЩЕМУ БАЙТУ
        MOV     A,M
        RAR                         ;СДВИНУТЬ ЦИКЛИЧЕСКИ БАЙТ МНОЖИМОГО
        MOV     M,A
        DCR     B
        JNZ     SRA1LP

        ;ЕСЛИ СЛЕДУЮЩИЙ РАЗРЯД МНОЖИМОГО РАВЕН 1,
        ; ТО ПРИБАВИТЬ МНОЖИТЕЛЬ К МАССИВУ HIPROD
        JNC     DECCNT              ;ПЕРЕЙТИ, ЕСЛИ СЛЕДУЮЩИЙ РАЗРЯД РАВЕН 0

        ;ПРИБАВИТЬ МНОЖИТЕЛЬ К МАССИВУ HIPROD
        PUSH    D                   ;СОХРАНИТЬ АДРЕС МНОЖИМОГО
        LHLD    MLIER               ;DE = АДРЕС МНОЖИТЕЛЯ
        XCHG
        LXI     H,HIPROD            ;HL = АДРЕС МАССИВА HIPROD
        MOV     B,C                 ;B = ДЛИНА В БАЙТАХ
        ANA     A                   ;ДЛЯ НАЧАЛА ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА

        ;ПРИБАВЛЯТЬ ПО БАЙТУ ЗА РАЗ
ADDLP:
        LDAX    D                   ;ВЗЯТЬ СЛЕДУЮЩИЙ БАЙТ МНОЖИТЕЛЯ
        ADC     M                   ;ПРИБАВИТЬ К МАССИВУ HIPROD
        MOV     M,A                 ;ЗАПОМНИТЬ В МАССИВЕ HIPROD НОВОЕ
                                    ; ЗНАЧЕНИЕ
        INX     D
        INX     H
        DCR     B
        JNZ     ADDLP               ;ПРОДОЛЖАТЬ ДЛЯ ВСЕХ БАЙТОВ МНОЖИТЕЛЯ
        POP     D                   ;ВОССТАНОВИТЬ АДРЕС МНОЖИМОГО

        ;УМЕНЬШИТЬ СЧЕТЧИК РАЗРЯДОВ, ВЫЙТИ, ЕСЛИ ВСЕ РАЗРЯДЫ ОБРАБОТАНЫ.
        ; ПРИ ЭТОМ ФЛАГ ПЕРЕНОСА ДОЛЖЕН ОСТАТЬСЯ БЕЗ ИЗМЕНЕНИЯ!
DECCNT:
        LDA     COUNT               ;ВЫЧЕСТЬ 1 ИЗ СЧЕТЧИКА РАЗРЯДОВ
        DCR     A
        STA     COUNT
        JNZ     LOOP                ;ПЕРЕЙТИ, ЕСЛИ МЛАДШИЙ БАЙТ СЧЕТЧИКА
                                    ; НЕ РАВЕН 0
        PUSH    PSW                 ;СОХРАНИТЬ ФЛАГ ПЕРЕНОСА
        LDA     COUNT+1             ;ВЗЯТЬ СТАРШИЙ БАЙТ СЧЕТЧИКА РАЗРЯДОВ
        ANA     A
        JZ      EXIT                ;ПЕРЕЙТИ, ЕСЛИ СЧЕТЧИК РАВЕН 0
        DCR     A                   ;УМЕНЬШИТЬ НА 1 СТАРШИЙ БАЙТ СЧЕТЧИКА
        STA     COUNT+1
        POP     PSW                 ;ВОССТАНОВИТЬ ФЛАГ ПЕРЕНОСА
        JMP     LOOP

EXIT:
        POP     PSW                 ;ПОЛУЧИТЬ ИЗ СТЕКА PSW
        RET
```

```
        ;ДАННЫЕ
COUNT:  DS      2                   ;ВРЕМЕННЫЕ ЯЧЕЙКИ ДЛЯ СЧЕТЧИКА ЦИКЛА
ENDHP:  DS      2                   ;АДРЕС ПОСЛЕДНЕГО БАЙТА
                                    ; МАССИВА HIPROD + 1
MLIER:  DS      2                   ;АДРЕС МНОЖИТЕЛЯ
HIPROD: DS      255                 ;БУФЕР ДЛЯ СТАРШЕЙ ЧАСТИ ПРОИЗВЕДЕНИЯ


;                                                               ;
;                                                               ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                       ;
;                                                               ;
;                                                               ;

SC6G:
        LXI     H,AY1               ;HL = БАЗОВЫЙ АДРЕС МНОЖИМОГО
        LXI     D,AY2               ;DE = БАЗОВЫЙ АДРЕС МНОЖИТЕЛЯ
        MVI     B,SZAYS             ;B = ДЛИНА ОПЕРАНДОВ В БАЙТАХ
        CALL    MPBMUL              ;УМНОЖИТЬ В ДВОИЧНОМ ВИДЕ С ПОВЫШЕННОЙ
                                    ; ТОЧНОСТЬЮ
                                    ;РЕЗУЛЬТАТ 12345H * 1234H = 14B60404H
                                    ; В ПАМЯТИ    AY1   = 04H
                                    ;             AY1+1 = 04H
                                    ;             AY1+2 = B6H
                                    ;             AY1+3 = 14H
                                    ;             AY1+4 = 00H
                                    ;             AY1+5 = 00H
                                    ;             AY1+6 = 00H

        JMP     SC6G

SZAYS   EQU     7                   ;ДЛИНА ОПЕРАНДОВ В БАЙТАХ
AY1:
        DB      045H
        DB      023H
        DB      001H
        DB      0
        DB      0
        DB      0
        DB      0

AY2:
        DB      034H
        DB      012H
        DB      0
        DB      0
        DB      0
        DB      0
        DB      0

        END
```

6Н. ДВОИЧНОЕ ДЕЛЕНИЕ ЧИСЕЛ С ПОВЫШЕННОЙ ТОЧНОСТЬЮ (MPBDIV)

Делятся два многобайтных беззнаковых двоичных числа. Оба числа хранятся в памяти таким образом, что их самые младшие по значению байты занимают самые младшие адреса. Частное замещает делимое; адрес младшего

по значению байта остатка находится в регистрах H и L. Длина чисел 255 байт или меньше. Если нет ошибок, флаг переноса очищается; при попытке деления на 0 флаг переноса устанавливается в 1, делимое остается без изменения, а остаток равен 0.

Процедура. С помощью обычного алгоритма сдвигов и вычитания осуществляется деление, при этом сдвигается частное и делимое и 1 помещается в делимое каждый раз, когда вычитание успешно. Результат пробного вычитания хранится в дополнительном буфере; если пробное вычитание успешно, то указатели этого буфера и буфера делимого просто переключаются (т. е. буферы меняются местами). Если определяется, что делитель равен нулю, немедленно осуществляется выход из программы и устанавливается флаг переноса. В противном случае флаг переноса очищается.

Используемые регистры: все.
Время выполнения: зависит от длины операндов и числа единичных разрядов в частном (требующих переключения буферов). Если среднее число единичных разрядов в частном составляет 4 на 1 байт, то время выполнения будет приблизительно равно $1240 * \text{LENGTH}^2 + 2046 * \text{LENGTH} + 515$ тактов, где LENGTH – число байтов в операндах.
Размер программы: 176 байт.
Память, необходимая для данных: 522 байта в любом месте ОЗУ. Это область для временного хранения старшей части делимого (255 байт, начиная с адреса HIDE1), результата пробного вычитания (255 байт, начиная с адреса HIDE2), базового адреса множимого (2 байта, начиная с адреса DVEND), базового адреса делителя (2 байта, начиная с адреса DVSOR), указателей для двух временных буферов для старшей части делимого (по 2 байта, начиная, соответственно, с адресов HDEPTR и ODEPTR), счетчика цикла (2 байта, начиная с адреса COUNT) и счетчика цикла вычитаний (1 байт по адресу SUBCNT).
Специальные случаи:
1) длина 0 вызывает немедленный выход с очищенным флагом переноса, частным, равным исходному делимому, и неопределенным остатком;
2) делитель, равный 0, вызывает выход с флагом переноса, установленным в 1, частным, равным исходному делимому, и остатком, равным 0.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес делимого в регистрах H и L.
Базовый адрес делителя в регистрах D и E.
Длина операндов в байтах в регистре B.

УСЛОВИЯ НА ВЫХОДЕ

Делимое заменяется делимым, деленным на делитель.
Если делитель ненулевой, то флаг переноса = 0 и результат нормальный.
Если делитель равен 0, то флаг переноса = 1, делимое остается без изменения, а остаток равен 0.
Остаток запоминается, начиная с младшего по значению байта по адресу, содержащемуся в регистрах H и L.

ПРИМЕР

1. Данные: длина операндов = 03 байта,
   делитель = $000F45_{16}$,
   делимое = $35A2F7_{16}$.

   Результат: делимое = 000383_{16},
   остаток (начиная с адреса в HL) = $0002A8_{16}$,
   флаг переноса равен 0, что показывает, что нет деления на 0.

```
;                                                                      ;
;                                                                      ;
;                                                                      ;
;                                                                      ;
;       ЗАГОЛОВОК:      ДВОИЧНОЕ ДЕЛЕНИЕ ЧИСЕЛ С ПОВЫШЕННОЙ ТОЧНОСТЬЮ  ;
;       ИМЯ:            MPBDIV                                         ;
;                                                                      ;
;                                                                      ;


;                                                                      ;
;       НАЗНАЧЕНИЕ:     ДЕЛИТ ДВА МАССИВА БАЙТОВ, СОДЕРЖАЩИХ ДВОИЧНЫЕ  ;
;                        ЧИСЛА                                         ;
;                       ДЕЛИМОЕ := ДЕЛИМОЕ / ДЕЛИТЕЛЬ                  ;
;                                                                      ;
;       ВХОД:           РЕГИСТРЫ H И L = БАЗОВЫЙ АДРЕС ДЕЛИМОГО        ;
;                       РЕГИСТРЫ D И E = БАЗОВЫЙ АДРЕС ДЕЛИТЕЛЯ        ;
;                       РЕГИСТР B = ДЛИНА ОПЕРАНДОВ В БАЙТАХ           ;
;                                                                      ;
;                         КАЖДЫЙ МАССИВ ДОЛЖЕН СОДЕРЖАТЬ ОДНО ДВОИЧНОЕ ;
;                         ЧИСЛО БЕЗ ЗНАКА С МАКСИМАЛЬНОЙ ДЛИНОЙ 255    ;
;                         БАЙТ. ARRAY[0] ЯВЛЯЕТСЯ МЛАДШИМ БАЙТОМ, А    ;
;                         ARRAY[LENGTH-1] - СТАРШИМ БАЙТОМ; ЗДЕСЬ      ;
;                         ARRAY - БАЗОВЫЙ АДРЕС МАССИВА, А LENGTH -    ;
;                         ДЛИНА.                                       ;
;                                                                      ;
;       ВЫХОД:          ДЕЛИМОЕ := ДЕЛИМОЕ / ДЕЛИТЕЛЬ                  ;
;                       РЕГИСТРЫ H И L = БАЗОВЫЙ АДРЕС ОСТАТКА         ;
;                       ЕСЛИ НЕТ ОШИБОК, ТО                            ;
;                         ФЛАГ ПЕРЕНОСА := 0                           ;
;                       ИНАЧЕ                                          ;
;                         ОШИБКА ДЕЛЕНИЯ НА 0                          ;
;                         ФЛАГ ПЕРЕНОСА := 1                           ;
;                         ДЕЛИМОЕ ОСТАЕТСЯ БЕЗ ИЗМЕНЕНИЯ               ;
;                         ОСТАТОК := 0                                 ;
;                                                                      ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ                                     ;
;                                                                      ;
;       ВРЕМЯ:          ЕСЛИ СЧИТАТЬ, ЧТО СРЕДНЕЕ ЧИСЛО РАЗРЯДОВ,      ;
;                       УСТАНОВЛЕННЫХ В 1, В ЧАСТНОМ РАВНО ДЛИНА/2,    ;
;                       ТО ВРЕМЯ ПРИБЛИЗИТЕЛЬНО РАВНО                  ;
;                        (1240 * ДЛИНА^2) + (2046 * ДЛИНА) + 515 ТАКТОВ ;
;                                                                      ;
;       РАЗМЕР:         ПРОГРАММА - 177 БАЙТ                           ;
;                       ДАННЫЕ    - 522 БАЙТА                          ;
;                                                                      ;
;                                                                      ;

MPBDIV:
        ;ПРОВЕРИТЬ ДЛИНУ ОПЕРАНДА, ЗАДАТЬ НАЧАЛЬНЫЕ ЗНАЧЕНИЯ УКАЗАТЕЛЕЙ
```

```
        MOV     A,B
        ORA     A               ;ДЛИНА МАССИВОВ = 0?
        JZ      OKEXIT          ;ЕСЛИ ДА, ТО ВЫЙТИ
        SHLD    DVEND           ;СОХРАНИТЬ БАЗОВЫЙ АДРЕС ДЕЛИМОГО
        XCHG
        SHLD    DVSOR           ;СОХРАНИТЬ БАЗОВЫЙ АДРЕС ДЕЛИТЕЛЯ
        MOV     C,B             ;C = ДЛИНА ОПЕРАНДОВ

        ;УСТАНОВИТЬ СЧЕТЧИК ЧИСЛА РАЗРЯДОВ В МАССИВАХ
        ; СЧЕТЧИК = (ДЛИНА * 8) + 1
        MOV     L,C             ;HL = ДЛИНА В БАЙТАХ
        MVI     H,0
        DAD     H               ;ДЛИНА * 2
        DAD     H               ;ДЛИНА * 4
        DAD     H               ;ДЛИНА * 8
        INX     H               ;ДЛИНА * 8 + 1
        SHLD    COUNT           ;СОХРАНИТЬ СЧЕТЧИК РАЗРЯДОВ

        ;ОБНУЛИТЬ ОБА МАССИВА СТАРШЕЙ ЧАСТИ ДЕЛИМОГО
        LXI     H,HIDE1         ;HL = АДРЕС МАССИВА HIDE1
        LXI     D,HIDE2         ;DE = АДРЕС МАССИВА HIDE2
        MOV     B,C             ;B = ДЛИНА
        SUB     A               ;ВЗЯТЬ 0 ДЛЯ ЗАПОЛНЕНИЯ
ZEROLP:
        MOV     M,A             ;ОБНУЛИТЬ ОБА МАССИВА ДЕЛИМОГО
        STAX    D
        INX     H
        INX     D
        DCR     B
        JNZ     ZEROLP

        ;СДЕЛАТЬ УКАЗАТЕЛЬ СТАРШЕЙ ЧАСТИ ДЕЛИМОГО РАВНЫМ АДРЕСУ
        ; МАССИВА HIDE1
        LXI     H,HIDE1
        SHLD    HDEPTR

        ;СДЕЛАТЬ ДРУГОЙ УКАЗАТЕЛЬ СТАРШЕЙ ЧАСТИ ДЕЛИМОГО РАВНЫМ
        ; АДРЕСУ МАССИВА HIDE2
        LXI     H,HIDE2
        SHLD    ODEPTR

        ;ИСПОЛЬЗУЯ ДЛЯ ВСЕХ БАЙТОВ ЛОГИЧЕСКУЮ ОПЕРАЦИЮ "ИЛИ",
        ; ПРОВЕРИТЬ, РАВЕН ЛИ ДЕЛИТЕЛЬ НУЛЮ
        LHLD    DVSOR           ;HL = АДРЕС ДЕЛИТЕЛЯ
        MOV     B,C             ;B = ДЛИНА В БАЙТАХ
        SUB     A               ;ОБНУЛИТЬ АККУМУЛЯТОР ДЛЯ ОПЕРАЦИИ "ИЛИ"
CHKOLP:
        ORA     M               ;ВЫПОЛНИТЬ "ИЛИ" ДЛЯ СЛЕДУЮЩЕГО БАЙТА
        INX     H               ;УВЕЛИЧИТЬ АДРЕС ДЛЯ СЛЕДУЮЩЕГО БАЙТА
        DCR     B
        JNZ     CHKOLP          ;ПРОДОЛЖАТЬ ДЛЯ ВСЕХ БАЙТОВ ДЕЛИТЕЛЯ
        ORA     A               ;УСТАНОВИТЬ ФЛАГИ С ПОМОЩЬЮ
                                ; ЛОГИЧЕСКОЙ ОПЕРАЦИИ "ИЛИ"
        JZ      EREXIT          ;ЕСЛИ ДЕЛИТЕЛЬ РАВЕН 0, ТО ВЫХОД ПО
                                ; ОШИБКЕ

        ;ВЫПОЛНИТЬ ДЕЛЕНИЕ, ИСПОЛЬЗУЯ АЛГОРИТМ ПРОБНОГО ВЫЧИТАНИЯ
        ORA     A               ;ДЛЯ НАЧАЛА ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА
```

```
LOOP:
        ;C = ДЛИНА
        ;DE = АДРЕС ДЕЛИТЕЛЯ
        ;ФЛАГ ПЕРЕНОСА = СЛЕДУЮЩИЙ РАЗРЯД ЧАСТНОГО
        ;СДВИНУТЬ ФЛАГ ПЕРЕНОСА В МАССИВ МЛАДШЕЙ ЧАСТИ ДЕЛИМОГО
        ; В КАЧЕСТВЕ СЛЕДУЮЩЕГО УКАЗАТЕЛЯ ЧАСТНОГО, А СТАРШИЙ РАЗРЯД
        ; МАССИВА МЛАДШЕЙ ЧАСТИ ДЕЛИМОГО - ВО ФЛАГ ПЕРЕНОСА
        MOV     B,C             ;B = ЧИСЛО БАЙТ, ПОДЛЕЖАЩИХ
                                ; ЦИКЛИЧЕСКОМУ СДВИГУ
        LHLD    DVEND           ;HL = АДРЕС ДЕЛИМОГО
SLLP1:
        MOV     A,M
        RAL                     ;СДВИНУТЬ ЦИКЛИЧЕСКИ ВЛЕВО БАЙТ ДЕЛИМОГО
        MOV     M,A
        INX     H               ;СЛЕДУЮЩИЙ БАЙТ
        DCR     B
        JNZ     SLLP1           ;ПРОДОЛЖАТЬ, ПОКА НЕ БУДУТ СДВИНУТЫ
                                ; ВСЕ БАЙТЫ

        ;УМЕНЬШИТЬ СЧЕТЧИКИ РАЗРЯДОВ И, ЕСЛИ ДЕЛЕНИЕ ВЫПОЛНЕНО, ВЫЙТИ
        ;ПРИ ЭТОМ ФЛАГ ПЕРЕНОСА НЕ ИЗМЕНЯЕТСЯ!
DECCNT:
        LDA     COUNT           ;ВЫЧЕСТЬ 1 ИЗ СЧЕТЧИКА
        DCR     A
        STA     COUNT
        JNZ     CONT            ;ПЕРЕЙТИ, ЕСЛИ МЛАДШИЙ БАЙТ СЧЕТЧИКА
                                ; НЕ РАВЕН НУЛЮ
        LDA     COUNT+1         ; ИНАЧЕ ЗАЕМ ИЗ СТАРШЕГО БАЙТА
        DCR     A
        STA     COUNT+1
        JM      OKEXIT          ;ВЫЙТИ, КОГДА СЧЕТЧИК СТАНЕТ
                                ; ОТРИЦАТЕЛЬНЫМ

        ;СДВИНУТЬ ФЛАГ ПЕРЕНОСА В МЛАДШИЙ БАЙТ СТАРШЕЙ ЧАСТИ ДЕЛИМОГО
CONT:
        LHLD    HDEPTR          ;HL = ТЕКУЩИЙ УКАЗАТЕЛЬ В СТАРШЕЙ ЧАСТИ
                                ; ДЕЛИМОГО
        MOV     B,C             ;B = ДЛИНА В БАЙТАХ
SLLP2:
        MOV     A,M
        RAL                     ;СДВИНУТЬ ЦИКЛИЧЕСКИ БАЙТ СТАРШЕЙ ЧАСТИ
                                ; ДЕЛИМОГО
        MOV     M,A
        INX     H               ;УВЕЛИЧИТЬ АДРЕС ДЛЯ СЛЕДУЮЩЕГО БАЙТА
        DCR     B
        JNZ     SLLP2           ;ПРОДОЛЖАТЬ, ПОКА НЕ БУДУТ СДВИНУТЫ
                                ; ВСЕ БАЙТЫ

        ;ВЫЧЕСТЬ ДЕЛИТЕЛЬ ИЗ СТАРШЕЙ ЧАСТИ ДЕЛИМОГО. ПОМЕСТИТЬ РАЗНОСТЬ
        ; В ДРУГОЙ МАССИВ СТАРШЕЙ ЧАСТИ ДЕЛИМОГО
        PUSH    B               ;СОХРАНИТЬ ДЛИНУ
        MOV     A,C
        STA     SUBCNT          ;SUBCNT = ДЛИНА В БАЙТАХ
        LHLD    ODEPTR
        MOV     C,L             ;BC = ДРУГОЕ ДЕЛИМОЕ
        MOV     B,H
        LHLD    HDEPTR
        XCHG                    ;DE = СТАРШАЯ ЧАСТЬ ДЕЛИМОГО
```

```
        LHLD    DVSOR               ;HL = ДЕЛИТЕЛЬ
        ORA     A                   ;ДЛЯ НАЧАЛА ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА
SUBLP:
        LDAX    D                   ;СЛЕДУЮЩИЙ БАЙТ СТАРШЕЙ ЧАСТИ ДЕЛИМОГО
        SBB     M                   ;ВЫЧЕСТЬ БАЙТ ДЕЛИТЕЛЯ
        STAX    B                   ;СОХРАНИТЬ В ДРУГОМ МАССИВЕ СТАРШЕЙ
                                    ; ЧАСТИ ДЕЛИМОГО
        INX     H                   ;УВЕЛИЧИТЬ УКАЗАТЕЛИ
        INX     D
        INX     B
        LDA     SUBCNT              ;УМЕНЬШИТЬ СЧЕТЧИК
        DCR     A
        STA     SUBCNT
        JNZ     SUBLP               ;ПРОДОЛЖАТЬ, ПОКА НЕ БУДЕТ ЗАКОНЧЕНО
                                    ; ВЫЧИТАНИЕ
        POP     B                   ;ВОССТАНОВИТЬ СЧЕТЧИК

        ;ЕСЛИ ФЛАГ ПЕРЕНОСА РАВЕН 1, ТО СТАРШАЯ ЧАСТЬ ДЕЛИМОГО МЕНЬШЕ
        ; ДЕЛИТЕЛЯ, И СЛЕДУЮЩИЙ РАЗРЯД ЧАСТНОГО РАВЕН 0.
        ; ЕСЛИ ФЛАГ ПЕРЕНОСА РАВЕН 0, ТО СЛЕДУЮЩИЙ РАЗРЯД ЧАСТНОГО РАВЕН
        ; 1, И МЫ ЗАМЕНЯЕМ ДЕЛИМОЕ НА ОСТАТОК, ПЕРЕКЛЮЧИВ УКАЗАТЕЛИ
        CMC                         ;ИНВЕРТИРОВАТЬ ФЛАГ ПЕРЕНОСА, ТАК ЧТОБЫ
                                    ; ОН СООТВЕТСТВОВАЛ СЛЕДУЮЩЕМУ РАЗРЯДУ
                                    ; ЧАСТНОГО
        JNC     LOOP                ;ПЕРЕЙТИ, ЕСЛИ СЛЕДУЮЩИЙ РАЗРЯД
                                    ; ЧАСТНОГО РАВЕН 0
        LHLD    HDEPTR              ;ИНАЧЕ ПОМЕНЯТЬ МЕСТАМИ УКАЗАТЕЛИ
        XCHG
        LHLD    ODEPTR
        SHLD    HDEPTR
        XCHG
        SHLD    ODEPTR

        ;ПРОДОЛЖИТЬ ДЛЯ СЛЕДУЮЩЕГО РАЗРЯДА ЧАСТНОГО, РАВНОГО 1
        ; (ФЛАГ ПЕРЕНОСА РАВЕН 1)
        JMP     LOOP

        ;УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА ПРИ ДЕЛЕНИИ НА НУЛЬ
EREXIT:
        STC                         ;УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА, РЕЗУЛЬТАТ
                                    ; НЕПРАВИЛЬНЫЙ
        JMP     EXIT

        ;ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА ПРИ ОТСУТСТВИИ ОШИБКИ
OKEXIT:
        ORA     A                   ;ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА,
                                    ; РЕЗУЛЬТАТ ПРАВИЛЬНЫЙ

        ;МАССИВ 1 ЯВЛЯЕТСЯ ЧАСТНЫМ
        ;HDEPTR СОДЕРЖИТ АДРЕС ОСТАТКА
EXIT:   LHLD    HDEPTR              ;HL = БАЗОВЫЙ АДРЕС ОСТАТКА
        RET

        ;ДАННЫЕ
DVEND:  DS      2                   ;АДРЕС ДЕЛИМОГО
DVSOR:  DS      2                   ;АДРЕС ДЕЛИТЕЛЯ
HDEPTR: DS      2                   ;АДРЕС ТЕКУЩЕГО МАССИВА СТАРШЕЙ ЧАСТИ
                                    ; ДЕЛИМОГО
```

```
ODEPTR: DS      2.              ;АДРЕС ДРУГОГО МАССИВА СТАРШЕЙ ЧАСТИ
                                ; ДЕЛИМОГО
COUNT:  DS      2               ;ВРЕМЕННЫЕ ЯЧЕЙКИ ДЛЯ СЧЕТЧИКА ЦИКЛА
SUBCNT: DS      1               ;СЧЕТЧИК ЦИКЛА ВЫЧИТАНИЙ
HIDE1:  DS      255             ;БУФЕР 1 СТАРШЕЙ ЧАСТИ ДЕЛИМОГО
HIDE2:  DS      255             ;БУФЕР 2 СТАРШЕЙ ЧАСТИ ДЕЛИМОГО

;                                                               ;
;                                                               ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                       ;
;                                                               ;
;                                                               ;

SC6H:
        LXI     H,AY1           ;HL = БАЗОВЫЙ АДРЕС ДЕЛИМОГО
        LXI     D,AY2           ;DE = БАЗОВЫЙ АДРЕС ДЕЛИТЕЛЯ
        MVI     B,SZAYS         ;B = ДЛИНА МАССИВОВ В БАЙТАХ
        CALL    MPBDIV          ;РАЗДЕЛИТЬ В ДВОИЧНОМ ВИДЕ С ПОВЫШЕННОЙ
                                ; ТОЧНОСТЬЮ
                                ;РЕЗУЛЬТАТ 14B60404H / 1234H = 12345H
                                ; В ПАМЯТИ AY1   = 45H
                                ;         AY1+1 = 23H
                                ;         AY1+2 = 01H
                                ;         AY1+3 = 00H
                                ;         AY1+4 = 00H
                                ;         AY1+5 = 00H
                                ;         AY1+6 = 00H

        JMP     SC6H

SZAYS   EQU     7               ;ДЛИНА МАССИВОВ В БАЙТАХ
AY1:
        DB      004H
        DB      004H
        DB      0B6H
        DB      014H
        DB      0
        DB      0
        DB      0

AY2:
        DB      034H
        DB      012H
        DB      0
        DB      0
        DB      0
        DB      0
        DB      0

        END
```

61. ДВОИЧНОЕ СРАВНЕНИЕ ЧИСЕЛ С ПОВЫШЕННОЙ ТОЧНОСТЬЮ (MPBCMP)

Сравниваются два многобайтных беззнаковых двоичных числа и соответствующим образом устанавливаются флаги переноса и нуля. Флаг нуля устанавливается в 1, если операнды равны, и в 0, если они не равны. Флаг переноса устанавливается в 1, если вычитаемое больше уменьшаемого; в против-

ном случае флаг переноса очищается. Таким образом, флаги устанавливаются так же, как если бы вычитаемое вычиталось из уменьшаемого.

Процедура. Сравниваются операнды побайтно, начиная с самых старших байтов и продолжая до тех пор, пока не будут найдены неравные соответствующие байты. Если все байты равны, осуществляется выход с флагом нуля, установленным в 1. Заметим, что при сравнении работа происходит с операндами, начиная с самых старших байтов, в то время как при вычитании (подпрограмма 6F) — начиная с самых младших.

Используемые регистры: все.
Время выполнения: 44 такта на проверяемый байт плюс приблизительно 45 тактов (8080) или 46 тактов на проверяемый байт плюс приблизительно 45 тактов (8085). Таким образом, сравнение продолжается, пока не будут найдены неравные соответствующие байты. Каждая пара проверяемых байтов требует 44 такта (8080) или 46 тактов (8085). Если все байты равны, то дополнительно к этому требуется 20 тактов.
Примеры:
1. Сравнение двух 6-байтных чисел, которые равны:
   $44 * 6 + 65 = 329$ тактов (8080),
   $44 * 6 + 65 = 341$ такт (8085).
2. Сравнение двух 8-байтных чисел, которые отличаются в байтах, следующих за самыми старшими по значению байтами:
   $44 * 2 + 45 = 133$ такта (8080),
   $46 * 2 + 45 = 137$ тактов (8085).
Размер программы: 54 байта.
Память, необходимая для данных: отсутствует.
Специальный случай: длина 0 вызывает немедленный выход с очищенным флагом переноса и флагом нуля, установленным в 1.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес уменьшаемого в регистрах H и L.
Базовый адрес вычитаемого в регистрах D и E.
Длина операндов в байтах в регистре B.

УСЛОВИЯ НА ВЫХОДЕ

Флаги устанавливаются так, как если бы вычитаемое вычиталось из уменьшаемого.
Флаг нуля = 1, если вычитаемое и уменьшаемое равны.
Флаг переноса устанавливается в 1, если вычитаемое больше уменьшаемого без учета знаков; флаг переноса равен 0, если вычитаемое меньше уменьшаемого или равно ему.

ПРИМЕРЫ

1. Данные: длина операндов = 6 байт,
   вычитаемое = $19D028A193EA_{16}$,
   уменьшаемое = $4E67BC15A266_{16}$.
   Результат: флаг нуля = 0 (операнды не равны),
   флаг переноса = 0 (вычитаемое не больше уменьшаемого).

2. Данные: длина операндов = 6 байт,
вычитаемое = $19D028A193EA_{16}$,
уменьшаемое = $19D028A193EA_{16}$,

Результат: флаг нуля = 1 (операнды равны),
флаг переноса = 0 (вычитаемое не больше уменьшаемого).

3. Даниые: длина операндов = 6 байт,
вычитаемое = $19D028A193EA_{16}$,
уменьшаемое = $0F37E5991D7C_{16}$.

Результат: флаг нуля = 0 (операнды не равны),
флаг переноса = 1 (вычитаемое больше уменьшаемого)

```
;                                                                  ;
;                                                                  ;
;                                                                  ;
;                                                                  ;
;       ЗАГОЛОВОК:      ДВОИЧНОЕ СРАВНЕНИЕ ЧИСЕЛ С ПОВЫШЕННОЙ ТОЧНОСТЬЮ ;
;       ИМЯ:            MPBCMP                                     ;
;                                                                  ;
;                                                                  ;


;                                                                  ;
;       НАЗНАЧЕНИЕ:     СРАВНИВАЕТ ДВА МАССИВА БАЙТОВ, СОДЕРЖАЩИХ  ;
;                        ДВОИЧНЫЕ ЧИСЛА, И УСТАНАВЛИВАЕТ ИЛИ ОЧИЩАЕТ ;
;                        ФЛАГИ ПЕРЕНОСА И НУЛЯ                     ;
;                                                                  ;
;       ВХОД:           РЕГИСТРЫ H И L = БАЗОВЫЙ АДРЕС УМЕНЬШАЕМОГО ;
;                       РЕГИСТРЫ D И E = БАЗОВЫЙ АДРЕС ВЫЧИТАЕМОГО ;
;                       РЕГИСТР B = ДЛИНА МАССИВОВ В БАЙТАХ        ;
;                                                                  ;
;                         КАЖДЫЙ МАССИВ ДОЛЖЕН СОДЕРЖАТЬ ОДНО ДВОИЧНОЕ ;
;                         ЧИСЛО БЕЗ ЗНАКА С МАКСИМАЛЬНОЙ ДЛИНОЙ 255 ;
;                         БАЙТ. ARRAY[0] ЯВЛЯЕТСЯ МЛАДШИМ БАЙТОМ, А ;
;                         ARRAY[LENGTH-1] - СТАРШИМ БАЙТОМ; ЗДЕСЬ  ;
;                         ARRAY - БАЗОВЫЙ АДРЕС МАССИВА, А LENGTH - ;
;                         ДЛИНА.                                   ;
;                                                                  ;
;       ВЫХОД:          ЕСЛИ УМЕНЬШАЕМОЕ = ВЫЧИТАЕМОМУ, ТО         ;
;                         C=0,Z=1                                  ;
;                       ЕСЛИ УМЕНЬШАЕМОЕ > ВЫЧИТАЕМОГО, ТО         ;
;                         C=0,Z=0                                  ;
;                       ЕСЛИ УМЕНЬШАЕМОЕ < ВЫЧИТАЕМОГО, ТО         ;
;                         C=1,Z=0                                  ;
;                                                                  ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ                                 ;
;                                                                  ;
;       ВРЕМЯ:          44 ТАКТА НА КАЖДЫЙ ПРОВЕРЯЕМЫЙ БАЙТ ПЛЮС   ;
;                        65 ТАКТОВ ДЛЯ 8080                        ;
;                       46 ТАКТОВ НА КАЖДЫЙ ПРОВЕРЯЕМЫЙ БАЙТ ПЛЮС  ;
;                        65 ТАКТОВ ДЛЯ 8085                        ;
;                                                                  ;
;       РАЗМЕР:         ПРОГРАММА - 20 БАЙТ                        ;
;                                                                  ;
;                                                                  ;

MPBCMP:
        ;ПРОВЕРИТЬ ДЛИНУ ОПЕРАНДОВ, УСТАНОВИТЬ УКАЗАТЕЛИ НА СТАРШИЕ
        ; БАЙТЫ
```

```
        MOV     A,B
        ORA     A               ;ДЛИНА МАССИВОВ = 0?
        RZ                      ;ДА, ВЫЙТИ С ФЛАГАМИ C=0, Z=1
        MOV     C,B             ;BC = ДЛИНА
        MVI     B,0
        DAD     B
        XCHG                    ;DE УКАЗЫВАЕТ НА КОНЕЦ УМЕНЬШАЕМОГО
        DAD     B               ;HL УКАЗЫВАЕТ НА КОНЕЦ ВЫЧИТАЕМОГО
        ORA     A               ;ПЕРЕД НАЧАЛОМ СРАВНЕНИЯ ОЧИСТИТЬ
                                ; ФЛАГ ПЕРЕНОСА

        ;ВЫЧИТАТЬ БАЙТЫ, НАЧИНАЯ С САМОГО СТАРШЕГО
        ;ЕСЛИ СООТВЕТСТВУЮЩИЕ БАЙТЫ НЕ СОВПАДАЮТ, ВЫЙТИ С
        ; УСТАНОВЛЕННЫМИ ФЛАГАМИ
LOOP:

        DCX     H               ;ПЕРЕЙТИ К МЕНЬШЕМУ ПО ЗНАЧЕНИЮ БАЙТУ
        DCX     D
        LDAX    D               ;ВЗЯТЬ СЛЕДУЮЩИЙ БАЙТ УМЕНЬШАЕМОГО
        SBB     M               ;ВЫЧЕСТЬ БАЙТ ВЫЧИТАЕМОГО
        RNZ                     ;ЕСЛИ НЕ РАВНЫ, ВЕРНУТЬСЯ
                                ; С УСТАНОВЛЕННЫМИ ФЛАГАМИ
        DCR     C
        JNZ     LOOP            ;ПРОДОЛЖАТЬ, ПОКА НЕ БУДЕТ ЗАКОНЧЕНО
                                ; СРАВНЕНИЕ
        RET                     ;РАВНЫ, ВЕРНУТЬСЯ С ФЛАГАМИ C=0, Z=1



;                                                                       ;
;                                                                       ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                               ;
;                                                                       ;
;                                                                       ;


SC6I:
        LXI     H,AY1           ;HL = БАЗОВЫЙ АДРЕС УМЕНЬШАЕМОГО
        LXI     D,AY2           ;DE = БАЗОВЫЙ АДРЕС ВЫЧИТАЕМОГО
        MVI     B,SZAYS         ;B = ДЛИНА МАССИВОВ В БАЙТАХ
        CALL    MPBCMP          ;СРАВНИТЬ В ДВОИЧНОМ ВИДЕ С ПОВЫШЕННОЙ
                                ; ТОЧНОСТЬЮ
                                ;В РЕЗУЛЬТАТЕ СРАВНЕНИЯ (7654321H,1234567H)
                                ; C=0, Z=0

        JMP     SC6I

SZAYS   EQU     7               ;ДЛИНА МАССИВОВ В БАЙТАХ
AY1:
        DB      021H
        DB      043H
        DB      065H
        DB      007H
        DB      0
        DB      0
        DB      0
```

```
AY2:
        DB      067H
        DB      045H
        DB      023H
        DB      001H
        DB      0
        DB      0
        DB      0

        END
```

6J. ДЕСЯТИЧНОЕ СЛОЖЕНИЕ ЧИСЕЛ С ПОВЫШЕННОЙ ТОЧНОСТЬЮ (MPDADD)

Складывает два многобайтных беззнаковых десятичных числа. Оба числа хранятся в памяти таким образом, что их самые младшие по значению байты занимают самые младшие адреса. Сумма замещает первое слагаемое. Длина чисел 255 байт или меньше.

Процедура. Сначала очищается флаг переноса, а затем складываются операнды по байту за раз (по две цифры), начиная с младших по значению цифр. Сумма замещает первое слагаемое. Длина 0 вызывает немедленный выход без сложения. Окончательный флаг переноса отражает сложение самых старших байтов.

Используемые регистры: все.
Время выполнения: 50 тактов на байт плюс 24 такта (8080) или 51 такт на байт плюс 21 такт (8085).
Размер программы: 14 байт.
Память, необходимая для данных: отсутствует.
Специальный случай: длина 0 вызывает немедленный выход, при этом операнд не изменяется, а флаг переноса очищается.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес первого слагаемого в регистрах H и L.
Базовый адрес второго слагаемого в регистрах D и E.
Длина операндов в байтах в регистре B.

УСЛОВИЯ НА ВЫХОДЕ

Первое слагаемое замещается первым слагаемым плюс второе слагаемое.

ПРИМЕР

1. Данные: длина операндов = 6 байт,
   первое слагаемое = 196028819315_{16},
   второе слагаемое = 293471605987_{16},
   Результат: первое слагаемое = 489500425302_{16},
   флаг переноса = 0.

```
;                                                                          ;
;                                                                          ;
;                                                                          ;
;                                                                          ;
;       ЗАГОЛОВОК:         ДЕСЯТИЧНОЕ СЛОЖЕНИЕ ЧИСЕЛ С ПОВЫШЕННОЙ          ;
;                          ТОЧНОСТЬЮ                                       ;
;       ИМЯ:               MPDADD                                          ;
;                                                                          ;
;                                                                          ;

;                                                                          ;
;       НАЗНАЧЕНИЕ:        СКЛАДЫВАЕТ ДВА МАССИВА БАЙТОВ В КОДЕ BCD        ;
;                          МАССИВ1 := МАССИВ1 + МАССИВ2                    ;
;                                                                          ;
;       ВХОД:              РЕГИСТРЫ H И L = БАЗОВЫЙ АДРЕС МАССИВА 1        ;
;                          РЕГИСТРЫ D И E = БАЗОВЫЙ АДРЕС МАССИВА 2        ;
;                          РЕГИСТР B = ДЛИНА МАССИВОВ В БАЙТАХ             ;
;                                                                          ;
;                            КАЖДЫЙ МАССИВ ДОЛЖЕН СОДЕРЖАТЬ ОДНО ЧИСЛО В   ;
;                            КОДЕ BCD БЕЗ ЗНАКА С МАКСИМАЛЬНОЙ ДЛИНОЙ      ;
;                            255 БАЙТ. ARRAY[0] ЯВЛЯЕТСЯ МЛАДШИМ БАЙТОМ, А ;
;                            ARRAY[LENGTH-1] - СТАРШИМ БАЙТОМ; ЗДЕСЬ       ;
;                            ARRAY - БАЗОВЫЙ АДРЕС МАССИВА, А LENGTH -     ;
;                            ДЛИНА.                                        ;
;                                                                          ;
;       ВЫХОД:             МАССИВ1 := МАССИВ1 + МАССИВ2                    ;
;                                                                          ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: A,B,DE,F,HL                                 ;
;                                                                          ;
;       ВРЕМЯ:             51 ТАКТ НА БАЙТ ПЛЮС 21 ТАКТ ДЛЯ 8085           ;
;                          50 ТАКТОВ НА БАЙТ ПЛЮС 24 ТАКТА ДЛЯ 8080        ;
;                                                                          ;
;       РАЗМЕР:            ПРОГРАММА - 14 БАЙТ                             ;
;                                                                          ;
;                                                                          ;

MPDADD:
        ;ПРОВЕРИТЬ ДЛИНУ МАССИВОВ НА 0, ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА
        MOV     A,B
        ANA     A                   ;ПРОВЕРИТЬ ДЛИНУ, ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА
        RZ                          ;ВОЗВРАТИТЬСЯ, ЕСЛИ ДЛИНА РАВНА 0

        ;СКЛАДЫВАТЬ ПО ДВЕ ЦИФРЫ ЗА РАЗ, ПРИНИМАЯ ВО ВНИМАНИЕ,
        ; ЧТО НАЧАЛЬНОЕ ЗНАЧЕНИЕ ФЛАГА ПЕРЕНОСА РАВНО 0
LOOP:
        LDAX    D
        ADC     M                   ;СЛОЖИТЬ СЛЕДУЮЩИЕ БАЙТЫ
        DAA                         ;ПРЕОБРАЗОВАТЬ В ДЕСЯТИЧНОЕ ЧИСЛО
        MOV     M,A                 ;ЗАПОМНИТЬ СУММУ
        INX     H                   ;ПОЛУЧИТЬ АДРЕСА СЛЕДУЮЩИХ БАЙТОВ
        INX     D
        DCR     B
        JNZ     LOOP                ;ПРОДОЛЖАТЬ, ПОКА НЕ БУДУТ СЛОЖЕНЫ
                                    ; ВСЕ БАЙТЫ
        RET


;                                                                          ;
```

```
;                                                                ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                        ;
;                                                                ;
;
;

SC6J:
        LXI     H,AY1           ;HL = БАЗОВЫЙ АДРЕС МАССИВА 1
        LXI     D,AY2           ;DE = БАЗОВЫЙ АДРЕС МАССИВА 2
        MVI     B,SZAYS         ;B = ДЛИНА МАССИВОВ В БАЙТАХ
        CALL    MPDADD          ;СЛОЖЕНИЕ ЧИСЕЛ В КОДЕ BCD С ПОВЫШЕННОЙ
                                ; ТОЧНОСТЬЮ. РЕЗУЛЬТАТ СЛОЖЕНИЯ
                                ; 1234567 + 1234567 = 2469134
                                ; В ПАМЯТИ AY1   = 34H
                                ;          AY1+1 = 91H
                                ;          AY1+2 = 46H
                                ;          AY1+3 = 02H
                                ;          AY1+4 = 00H
                                ;          AY1+5 = 00H
                                ;          AY1+6 = 00H

        JMP     SC6J

SZAYS   EQU     7               ;ДЛИНА МАССИВОВ В БАЙТАХ
AY1:
        DB      067H
        DB      045H
        DB      023H
        DB      001H
        DB      0
        DB      0
        DB      0

AY2:
        DB      067H
        DB      045H
        DB      023H
        DB      001H
        DB      0
        DB      0
        DB      0

        END
```

6К. ДЕСЯТИЧНОЕ ВЫЧИТАНИЕ ЧИСЕЛ С ПОВЫШЕННОЙ ТОЧНОСТЬЮ (MPDSUB)

Вычитаются два многобайтных десятичных числа. Оба числа хранятся в памяти таким образом, что их самые младшие по значению байты занимают самые младшие адреса. Разность замещает уменьшаемое. Длина чисел 255 байт или меньше.

Процедура. Сначала очищается флаг переноса, а затем вычитаемое вычитается из уменьшаемого по одному байту (две цифры) за раз, начиная с самых младших по значению цифр. Длина 0 вызывает немедленный выход без вычитания. Окончательный флаг переноса является инвертированным заемом и отражает вычитание самых младших по значению цифр.

Используемые регистры: все.
Время выполнения: 73 такта на байт плюс 32 такта (8080) или 73 такта на байт плюс 29 тактов (8085).
Размер программы: 22 байта.
Память, необходимая для данных: отсутствует.
Специальный случай: длина 0 вызывает немедленный выход, при этом уменьшаемое не изменяется (т. е. разность равна уменьшаемому). Флаг переноса очищается.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес уменьшаемого в регистрах H и L.
Базовый адрес вычитаемого в регистрах D и E.
Длина операндов в регистре B.

УСЛОВИЯ НА ВЫХОДЕ

Уменьшаемое замещается уменьшаемым минус вычитаемое.

ПРИМЕР

1. Данные: длина операндов (в байтах) = 6,
уменьшаемое = 293471605987_{16},
вычитаемое = 196028819315_{16}.
Результат: уменьшаемое = 097442786672_{16},
флаг переноса = 1, так как не требуется заема.

```
;                                                                    ;
;                                                                    ;
;                                                                    ;
;                                                                    ;
;       ЗАГОЛОВОК:      ДЕСЯТИЧНОЕ ВЫЧИТАНИЕ ЧИСЕЛ С ПОВЫШЕННОЙ      ;
;                       ТОЧНОСТЬЮ                                    ;
;       ИМЯ:            MPDSUB                                       ;
;                                                                    ;
;                                                                    ;


;                                                                    ;
;       НАЗНАЧЕНИЕ:     ВЫЧИТАЕТ ДВА МАССИВА БАЙТОВ В КОДЕ BCD       ;
;                       УМЕНЬШАЕМОЕ := УМЕНЬШАЕМОЕ - ВЫЧИТАЕМОЕ      ;
;                                                                    ;
;       ВХОД:           РЕГИСТРЫ H И L = БАЗОВЫЙ АДРЕС УМЕНЬШАЕМОГО  ;
;                       РЕГИСТРЫ D И E = БАЗОВЫЙ АДРЕС ВЫЧИТАЕМОГО   ;
;                       РЕГИСТР B = ДЛИНА МАССИВОВ В БАЙТАХ          ;
;                                                                    ;
;                         КАЖДЫЙ МАССИВ ДОЛЖЕН СОДЕРЖАТЬ ОДНО ЧИСЛО В ;
;                         КОДЕ BCD БЕЗ ЗНАКА С МАКСИМАЛЬНОЙ ДЛИНОЙ  ;
;                         255 БАЙТ. ARRAY[0] ЯВЛЯЕТСЯ МЛАДШИМ БАЙТОМ, А ;
;                         ARRAY[LENGTH-1] - СТАРШИМ БАЙТОМ; ЗДЕСЬ    ;
;                         ARRAY - БАЗОВЫЙ АДРЕС МАССИВА, А LENGTH -  ;
;                         ДЛИНА.                                     ;
;                                                                    ;
;       ВЫХОД:          УМЕНЬШАЕМОЕ := УМЕНЬШАЕМОЕ - ВЫЧИТАЕМОЕ      ;
;                                                                    ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ                                   ;
;                                                                    ;
```

```
;       ВРЕМЯ:          73 ТАКТА НА БАЙТ ПЛЮС 29 ТАКТОВ ДЛЯ 8085        ;
;                       73 ТАКТА НА БАЙТ ПЛЮС 32 ТАКТА ДЛЯ 8080         ;
;                                                                       ;
;       РАЗМЕР:         ПРОГРАММА - 22 БАЙТА                            ;
;                                                                       ;
;                                                                       ;

MPDSUB:
        ;ПРОВЕРИТЬ ДЛИНУ МАССИВОВ НА 0, ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА
        MOV     A,B
        ORA     A               ;ПРОВЕРИТЬ ДЛИНУ, ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА
        RZ                      ;ВЫЙТИ, ЕСЛИ ДЛИНА РАВНА 0
        CMC                     ;ДЛЯ ПОЛУЧЕНИЯ ДОПОЛНЕНИЯ ДО 10
                                ; УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА
        XCHG                    ;HL = УМЕНЬШАЕМОЕ
                                ;DE = ВЫЧИТАЕМОЕ

        ;ВЫЧИТАТЬ ОПЕРАНДЫ ПО 2 ЦИФРЫ ЗА РАЗ, ПРИБАВЛЯЯ ДОПОЛНЕНИЕ
        ; ВЫЧИТАЕМОГО ДО ДЕСЯТИ К УМЕНЬШАЕМОМУ
        ;В АРИФМЕТИЧЕСКИХ ОПЕРАЦИЯХ С ДОПОЛНЕНИЯМИ ДО ДЕСЯТИ ФЛАГ
        ; ПЕРЕНОСА ЯВЛЯЕТСЯ ИНВЕРТИРОВАННЫМ ЗАЕМОМ
        ;ЗАМЕТИМ, ЧТО КОМАНДА DAA ВЫПОЛНЯЕТСЯ ТОЛЬКО ПОСЛЕ КОМАНД СЛОЖЕНИЯ
        ;БАЙТ ДОПОЛНЕНИЯ ДО ДЕСЯТИ РАВЕН ШЕСТНАДЦАТЕРИЧНОМУ ЧИСЛУ 99
        ; + ИНВЕРТИРОВАННЫЙ ЗАЕМ - БАЙТ ВЫЧИТАЕМОГО. РЕЗУЛЬТАТ ВСЕГДА
        ; НЕОТРИЦАТЕЛЬНЫЙ, А ФЛАГИ ПЕРЕНОСА И ВСПОМОГАТЕЛЬНОГО ПЕРЕНОСА
        ; ВСЕГДА РАВНЫ 0, ТАК ЧТО ПРИ ОПЕРАЦИЯХ С КОДАМИ BCD НЕ ВОЗНИКАЕТ
        ; ПРОБЛЕМ.
LOOP:
        MVI     A,99H           ;СФОРМИРОВАТЬ БАЙТ, СОДЕРЖАЩИЙ
        ACI     0               ; ДОПОЛНЕНИЕ ВЫЧИТАЕМОГО ДО 10
        SUB     M
        MOV     C,A
        LDAX    D               ;ВЗЯТЬ УМЕНЬШАЕМОЕ
        ADD     C               ;ПРИБАВИТЬ ДОПОЛНЕНИЕ УМЕНЬШАЕМОГО ДО 10
        DAA                     ;ПРЕОБРАЗОВАТЬ В ДЕСЯТИЧНОЕ ЧИСЛО
        STAX    D               ;ЗАПОМНИТЬ РАЗНОСТЬ
        INX     H               ;ПОЛУЧИТЬ АДРЕСА СЛЕДУЮЩИХ БАЙТОВ
        INX     D
        DCR     B
        JNZ     LOOP            ;ПРОДОЛЖАТЬ, ПОКА НЕ БУДУТ ВЫЧТЕНЫ
                                ; ВСЕ БАЙТЫ
        RET


;                                                                       ;
;                                                                       ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                               ;
;                                                                       ;
;                                                                       ;

SC6K:
        LXI     H,AY1           ;HL = БАЗОВЫЙ АДРЕС УМЕНЬШАЕМОГО
        LXI     D,AY2           ;DE = БАЗОВЫЙ АДРЕС ВЫЧИТАЕМОГО
        MVI     B,SZAYS         ;B = ДЛИНА МАССИВОВ В БАЙТАХ
        CALL    MPDSUB          ;ВЫЧИТАНИЕ ЧИСЕЛ В КОДЕ BCD С ПОВЫШЕННОЙ
                                ; ТОЧНОСТЬЮ. РЕЗУЛЬТАТ ВЫЧИТАНИЯ
                                ; 2469134 + 1234567 = 1234567
                                ; В ПАМЯТИ AY1    = 67H
```

```
                                ;           AY1+1 = 45H
                                ;           AY1+2 = 23H
                                ;           AY1+3 = 01H
                                ;           AY1+4 = 00H
                                ;           AY1+5 = 00H
                                ;           AY1+6 = 00H

        JMP     SC6K

SZAYS   EQU     7                       ;ДЛИНА МАССИВОВ В БАЙТАХ
AY1:
        DB      034H
        DB      091H
        DB      046H
        DB      002H
        DB      0
        DB      0
        DB      0

AY2:
        DB      067H
        DB      045H
        DB      023H
        DB      001H
        DB      0
        DB      0
        DB      0

        END
```

6L. ДЕСЯТИЧНОЕ УМНОЖЕНИЕ ЧИСЕЛ С ПОВЫШЕННОЙ ТОЧНОСТЬЮ (MPDMUL)

Умножаются два многобайтных беззнаковых десятичных числа. Оба числа хранятся в памяти таким образом, что их самые младшие по значению байты занимают самые младшие адреса. Произведение замещает множимое. Длина чисел 255 байт или меньше. Для совместимости с другими десятичными операциями с повышенной точностью возвращаются только младшие по значению байты произведения.

Процедура. Каждая цифра множимого рассматривается в отдельности. С помощью маски выделяется цифра, сдвигается (если цифра находится в верхней половине байта), а затем используется в качестве счетчика для определения того, сколько раз прибавить множитель к промежуточному произведению. Младшая по значению цифра промежуточного произведения сохраняется в качестве следующей цифры полного произведения, а промежуточное произведение сдвигается вправо на четыре разряда. Для того чтобы определить, с какой цифрой байта происходит в данный момент работа, верхней или нижней, используется флаг. Длина 0 вызывает выход без умножения.

Используемые регистры: все.
Время выполнения: зависит от длины операндов и значений цифр множимого (так как эти цифры определяют, сколько раз множитель должен быть прибавлен к промежуточному произведению). Если в среднем цифры множимого имеют значение 5, то время выполнения приблизительно равно 726 * LENGTH² + 1603 *

$*$ LENGTH + 151 такт (8080) или 730 $*$ LENGTH^2 + 1503 $*$ LENGTH + 139 тактов (8085), где LENGTH – число байтов в операндах.

Размер программы: 195 байт.

Память, необходимая для данных: 520 байт в любом месте ОЗУ. Это область для временного хранения старших байтов промежуточного произведения (255 байт, начиная с адреса PROD), множимого (255 байт, начиная с адреса MCAND), длины массивов (1 байт по адресу LEN), счетчика цифр, указывающего на верхнюю или нижнюю цифру (1 байт по адресу DCNT), счетчика цикла (1 байт по адресу LPCNT), байта переполнения (1 байт по адресу OVERFLW), указателей для множимого и множителя (по 2 байта, начиная с адресов MCADR и MPADR соответственно) и следующего байта множимого (1 байт по адресу NBYTE).

Специальный случай: длина 0 вызывает немедленный выход, при этом множимое не изменяется. Старшие по значению байты произведения (начинающиеся с адреса PROD) имеют неопределенное значение.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес множимого в регистрах H и L.
Базовый адрес множителя в регистрах D и E.
Длина операндов в байтах в регистре B.

УСЛОВИЯ НА ВЫХОДЕ

Множимое заменяется множимым, умноженным на множитель.

ПРИМЕР

1. Данные: длина операндов = 04 байта,
   множитель = 00003518_{16},
   множимое = 00006294_{16}.

   Результат: множимое = 22142292_{16}.

Заметим, что для совместимости с другими десятичными арифметическими операциями MPDMUL возвращает только младшие по значению байты произведения (т. е. число байтов в множимом и множителе). Более старшие по значению байты произведения доступны, начиная с их младших по значению цифр по адресу PROD. У пользователя может появиться потребность в проверке этих байтов на возможность переполнения или для расширения операндов с помощью дополнительных нулей.

```
;                                                              ;
;                                                              ;
;                                                              ;
;                                                              ;
;       ЗАГОЛОВОК:      ДЕСЯТИЧНОЕ УМНОЖЕНИЕ ЧИСЕЛ С ПОВЫШЕННОЙ ;
;                       ТОЧНОСТЬЮ                              ;
;       ИМЯ:            MPDMUL                                 ;
;                                                              ;
;                                                              ;


;                                                              ;
;       НАЗНАЧЕНИЕ:     УМНОЖАЕТ ДВА МАССИВА БАЙТ В КОДЕ BCD   ;
;                       МНОЖИМОЕ := МНОЖИМОЕ * МНОЖИТЕЛЬ       ;
;                                                              ;
;       ВХОД:           РЕГИСТРЫ H И L = БАЗОВЫЙ АДРЕС МНОЖИМОГО ;
```

```
;                       РЕГИСТРЫ D И E = БАЗОВЫЙ АДРЕС МНОЖИТЕЛЯ       ;
;                       РЕГИСТР B = ДЛИНА МАССИВОВ В БАЙТАХ            ;
;                                                                      ;
;                         КАЖДЫЙ МАССИВ ДОЛЖЕН СОДЕРЖАТЬ ОДНО ЧИСЛО В  ;
;                         КОДЕ BCD БЕЗ ЗНАКА С МАКСИМАЛЬНОЙ ДЛИНОЙ     ;
;                         255 БАЙТ. ARRAY[0] ЯВЛЯЕТСЯ МЛАДШИМ БАЙТОМ, А ;
;                         ARRAY[LENGTH-1] - СТАРШИМ БАЙТОМ; ЗДЕСЬ      ;
;                         ARRAY - БАЗОВЫЙ АДРЕС МАССИВА, А LENGTH -    ;
;                         ДЛИНА.                                       ;
;                                                                      ;
;       ВЫХОД:          МНОЖИМОЕ := МНОЖИМОЕ * МНОЖИТЕЛЬ               ;
;                                                                      ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ                                     ;
;                                                                      ;
;       ВРЕМЯ:          ЕСЛИ ПРИНЯТЬ, ЧТО СРЕДНЕЕ ЗНАЧЕНИЕ ЦИФРЫ       ;
;                       МНОЖИМОГО РАВНО 5,ТО ВРЕМЯ ПРИБЛИЗИТЕЛЬНО РАВНО ;
;                        (726 * ДЛИНА^2) + (1603 * ДЛИНА) + 151 ТАКТ   ;
;                        ДЛЯ 8080                                      ;
;                        (730 * ДЛИНА^2) + (1503 * ДЛИНА) + 139 ТАКТОВ ;
;                        ДЛЯ 8085                                      ;
;                                                                      ;
;       РАЗМЕР:         ПРОГРАММА - 195 БАЙТ                           ;
;                       ДАННЫЕ    - 520 БАЙТ                           ;
;                                                                      ;
;                                                                      ;

MPDMUL:
        ;ИНИЦИАЛИЗИРОВАТЬ СЧЕТЧИКИ И УКАЗАТЕЛЬ
        MOV     A,B             ;ПРОВЕРИТЬ ДЛИНУ ОПЕРАНДОВ
        ORA     A
        RZ                      ;ВЫЙТИ, ЕСЛИ ДЛИНА РАВНА 0
        STA     LEN             ;СОХРАНИТЬ ДЛИНУ
        STA     LPCNT           ;СЧЕТЧИК ЦИКЛА = ДЛИНА В БАЙТАХ
        SHLD    MCADR           ;СОХРАНИТЬ АДРЕС МНОЖИМОГО
        XCHG
        SHLD    MPADR           ;СОХРАНИТЬ АДРЕС МНОЖИТЕЛЯ

        ;СОХРАНИТЬ МНОЖИМОЕ ВО ВРЕМЕННОМ БУФЕРЕ (MCAND)
        ;ОЧИСТИТЬ ПРОМЕЖУТОЧНОЕ ПРОИЗВЕДЕНИЕ, СОДЕРЖАЩЕЕ НУЛИ В
        ; СТАРШИХ БАЙТАХ, НАЧИНАЮЩИХСЯ С PROD, И МНОЖИМОЕ В МЛАДШИХ
        ; БАЙТАХ
        LXI     H,MCAND
        SHLD    NBYTE           ;СЛЕДУЮЩИЙ БАЙТ = МЛАДШИЙ БАЙТ
                                ; МНОЖИМОГО
        XCHG                    ;HL = АДРЕС МНОЖИМОГО
                                ;DE = АДРЕС ВРЕМЕННОГО МНОЖИМОГО
                                ;B = ДЛИНА ОПЕРАНДОВ В БАЙТАХ
        MVI     C,0             ;ВЗЯТЬ 0 ДЛЯ ЗАПОЛНЕНИЯ ПРОМЕЖУТОЧНОГО
                                ; ПРОИЗВЕДЕНИЯ
INITLP0:
        MOV     A,M             ;ВЗЯТЬ СЛЕДУЮЩИЙ БАЙТ МНОЖИМОГО
        STAX    D               ;ЗАПОМНИТЬ ВО ВРЕМЕННОМ БУФЕРЕ
        MOV     M,C             ;ОЧИСТИТЬ БАЙТ МНОЖИМОГО
        INX     D
        INX     H
        DCR     B
        JNZ     INITLP0         ;ПРОДОЛЖАТЬ, ПОКА НЕ БУДЕТ ВЫПОЛНЕНО
```

```
        ;ОЧИСТИТЬ СТАРШИЕ БАЙТЫ ПРОМЕЖУТОЧНОГО ПРОИЗВЕДЕНИЯ
        LXI     H,PROD              ;HL = БАЗОВЫЙ АДРЕС ПРОИЗВЕДЕНИЯ
        LDA     LEN
        MOV     B,A                 ;B = ДЛИНА ОПЕРАНДОВ В БАЙТАХ
                                    ;C ВСЕ ЕЩЕ СОДЕРЖИТ 0 ДЛЯ ЗАПОЛНЕНИЯ
INITLP1:
        MOV     M,C                 ;ОБНУЛИТЬ БАЙТ ПРОИЗВЕДЕНИЯ
        INX     H
        DCR     B
        JNZ     INITLP1             ;ПРОДОЛЖАТЬ, ПОКА НЕ БУДУТ ОБНУЛЕНЫ
                                    ; ВСЕ БАЙТЫ

        ;ЦИКЛ ДЛЯ ВСЕХ БАЙТОВ МНОЖИМОГО
LOOP:
        MVI     A,1
        STA     DCNT                ;НАЧАТЬ С МЛАДШЕЙ ЦИФРЫ

        ;ЦИКЛ ДЛЯ ДВУХ ЦИФР НА БАЙТ
        ; ДЛЯ МЛАДШЕЙ ЦИФРЫ DCNT = 1
        ; ДЛЯ СТАРШЕЙ ЦИФРЫ DCNT = 0
DLOOP:
        SUB     A                   ;ОЧИСТИТЬ БАЙТ ПЕРЕПОЛНЕНИЯ
        STA     OVRFLW
        LDA     DCNT
        ORA     A                   ;ПРОВЕРИТЬ, МЛАДШАЯ ЛИ ЦИФРА (Z = 0)
        LHLD    NBYTE               ;ВЗЯТЬ СЛЕДУЮЩИЙ БАЙТ
        MOV     A,M
        JNZ     DLOOP1              ;ПЕРЕЙТИ, ЕСЛИ ЦИФРА МЛАДШАЯ
        RRC                         ;СДВИНУТЬ СТАРШУЮ ЦИФРУ ВПРАВО
        RRC                         ; НА 4 РАЗРЯДА
        RRC
        RRC

DLOOP1:
        ANI     0FH                 ;ВЫДЕЛИТЬ СЛЕДУЮЩУЮ ЦИФРУ
        JZ      SDIGIT              ;ПЕРЕЙТИ, ЕСЛИ СЛЕДУЮЩАЯ ЦИФРА НУЛЬ
        MOV     C,A                 ;C = СЛЕДУЮЩАЯ ЦИФРА

        ;ПРИБАВИТЬ МНОЖИТЕЛЬ К ПРОИЗВЕДЕНИЮ NDIGIT РАЗ
ADDLP:
        LHLD    MPADR               ;HL = БАЗОВЫЙ АДРЕС МНОЖИТЕЛЯ
        LXI     D,PROD              ;DE = БАЗОВЫЙ АДРЕС ПРОИЗВЕДЕНИЯ
        LDA     LEN
        MOV     B,A                 ;B = ДЛИНА
        ORA     A                   ;ДЛЯ НАЧАЛА ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА

        ;ПРИБАВЛЯТЬ МНОЖИТЕЛЬ К ПРОИЗВЕДЕНИЮ ПО 1 БАЙТУ ЗА РАЗ
INNER:
        LDAX    D                   ;ВЗЯТЬ СЛЕДУЮЩИЙ БАЙТ ПРОИЗВЕДЕНИЯ
        ADC     M                   ;ПРИБАВИТЬ БАЙТ МНОЖИТЕЛЯ
        DAA                         ;КОРРЕКТИРОВАТЬ В ДЕСЯТИЧНЫЙ ВИД
        STAX    D                   ;ЗАПОМНИТЬ СУММУ В ПРОИЗВЕДЕНИИ
        INX     H
        INX     D
        DCR     B
        JNZ     INNER               ;ПРОДОЛЖАТЬ, ПОКА НЕ БУДУТ ПРИБАВЛЕНЫ
                                    ; ВСЕ БАЙТЫ
        JNC     DECND               ;ПЕРЕЙТИ, ЕСЛИ ПОСЛЕ СЛОЖЕНИЯ НЕТ
```

```
                                        ; ПЕРЕПОЛНЕНИЯ
        LXI     H,OVRFLW                ;ИНАЧЕ УВЕЛИЧИТЬ БАЙТ ПЕРЕПОЛНЕНИЯ
        INR     M
DECND:
        DCR     C
        JNZ     ADDLP                   ;ПРОДОЛЖАТЬ, ПОКА ЦИФРА НЕ БУДЕТ
                                        ; РАВНА 0

        ;ЗАПОМНИТЬ МЛАДШУЮ ЦИФРУ ПРОИЗВЕДЕНИЯ КАК СЛЕДУЮЩУЮ ЦИФРУ
        ; МНОЖИМОГО
SDIGIT:
        LDA     PROD                    ;ВЗЯТЬ СЛЕДУЮЩИЙ БАЙТ ПРОИЗВЕДЕНИЯ
        ANI     0FH
        MOV     B,A                     ;СОХРАНИТЬ В РЕГИСТРЕ B
        LDA     DCNT
        ORA     A                       ;ПРОВЕРИТЬ, МЛАДШАЯ ЛИ ЦИФРА (Z = 0)
        MOV     A,B                     ;A = СЛЕДУЮЩАЯ ЦИФРА
        JNZ     SD1                     ;ПЕРЕЙТИ, ЕСЛИ ОБРАБАТЫВАЕТСЯ МЛАДШАЯ
                                        ; ЦИФРА
        RRC                             ;ИНАЧЕ ПЕРЕСЛАТЬ СТАРШУЮ ЦИФРУ В МЛАДШУЮ
        RRC
        RRC
        RRC

SD1:
        LHLD    MCADR                   ;ПОМЕСТИТЬ СЛЕДУЮЩУЮ ЦИФРУ В МНОЖИМОЕ
        ORA     M
        MOV     M,A

        ;СДВИНУТЬ ПРОИЗВЕДЕНИЕ ВПРАВО НА 1 ЦИФРУ (4 РАЗРЯДА)
        LDA     LEN
        MOV     B,A                     ;B = ДЛИНА
        MOV     E,A
        MVI     D,0
        LXI     H,PROD
        DAD     D                       ;HL УКАЗЫВАЕТ НА ЯЧЕЙКУ ЗА КОНЦОМ PROD
        LDA     OVRFLW
        MOV     D,A                     ;D = БАЙТ ПЕРЕПОЛНЕНИЯ
SHFTLP:
        DCX     H                       ;УМЕНЬШИТЬ, ПОЛУЧИТЬ АДРЕС СЛЕДУЮЩЕГО
                                        ; БАЙТА
        MOV     A,M
        MOV     C,A                     ;СОХРАНИТЬ МЛАДШУЮ ЦИФРУ В РЕГИСТРЕ C
                                        ; ДЛЯ ПОСЛЕДУЮЩЕГО ИСПОЛЬЗОВАНИЯ
        ANI     0F0H                    ;ОЧИСТИТЬ МЛАДШУЮ ЦИФРУ
        ORA     D                       ;СФОРМИРОВАТЬ ПРОИЗВЕДЕНИЕ И ПЕРЕПОЛНЕНИЕ
        RRC                             ;ПОМЕНЯТЬ ЦИФРЫ МЕСТАМИ, ТАК ЧТОБЫ
        RRC                             ; СТАРШАЯ ЦИФРА БЫЛА ИЗ ПЕРЕПОЛНЕНИЯ,
        RRC                             ; А МЛАДШАЯ - ИЗ ПРОИЗВЕДЕНИЯ
        RRC
        MOV     M,A                     ;ЗАПОМНИТЬ ЦИФРЫ В ПРОИЗВЕДЕНИИ
        MOV     A,C                     ;ВЗЯТЬ НАЗАД МЛАДШУЮ ЦИФРУ
        ANI     0FH
        MOV     D,A                     ;СОХРАНИТЬ ЕЕ ДЛЯ СЛЕДУЮЩЕГО ЦИКЛА
        DCR     B
        JNZ     SHFTLP                  ;ПРОДОЛЖАТЬ, ПОКА НЕ БУДЕТ ВЫПОЛНЕНО
```

```
        ;ПРОВЕРИТЬ, ОБЕ ЛИ ЦИФРЫ ТЕКУЩЕГО БАЙТА ОБРАБОТАНЫ
        LXI     H,DCNT          ;МЛАДШАЯ ЦИФРА?
        DCR     M
        JZ      DLOOP           ;ДА, ПЕРЕЙТИ К СТАРШЕЙ ЦИФРЕ БАЙТА

        ;УВЕЛИЧИТЬ УКАЗАТЕЛИ ДЛЯ СЛЕДУЮЩЕГО БАЙТА И ПРОВЕРИТЬ НА
        ; ОКОНЧАНИЕ РАБОТЫ
        LHLD    NBYTE           ;УВЕЛИЧИТЬ ДЛЯ СЛЕДУЮЩЕГО БАЙТА
        INX     H               ; МНОЖИМОГО
        SHLD    NBYTE
        LHLD    MCADR           ;УВЕЛИЧИТЬ ДЛЯ СЛЕДУЮЩЕГО БАЙТА
        INX     H               ; РЕЗУЛЬТАТА
        SHLD    MCADR
        LXI     H,LPCNT         ;УМЕНЬШИТЬ СЧЕТЧИК ЦИКЛА
        DCR     M
        JNZ     LOOP
EXIT:
        RET

        ;ДАННЫЕ
LEN:            DS      1       ;ДЛИНА МАССИВОВ В БАЙТАХ
DCNT:           DS      1       ;СЧЕТЧИК ЦИФР ДЛЯ БАЙТОВ В МАССИВАХ
LPCNT:          DS      1       ;СЧЕТЧИК ЦИКЛА
OVRFLW:         DS      1       ;БАЙТ ПЕРЕПОЛНЕНИЯ
MCADR:          DS      2       ;УКАЗАТЕЛЬ ДЛЯ МНОЖИМОГО
MPADR:          DS      2       ;УКАЗАТЕЛЬ ДЛЯ МНОЖИТЕЛЯ
NBYTE:          DS      2       ;СЛЕДУЮЩИЙ БАЙТ МНОЖИМОГО
PROD:           DS      255     ;БУФЕР ПРОИЗВЕДЕНИЯ
MCAND:          DS      255     ;БУФЕР МНОЖИМОГО


;                                                               ;
;                                                               ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                       ;
;                                                               ;
;                                                               ;

SC6L:
        LXI     H,AY1           ;БАЗОВЫЙ АДРЕС МНОЖИМОГО
        LXI     D,AY2           ;БАЗОВЫЙ АДРЕС МНОЖИТЕЛЯ
        MVI     B,SZAYS         ;ДЛИНА МАССИВОВ В БАЙТАХ
        CALL    MPDMUL          ;УМНОЖЕНИЕ ЧИСЕЛ В КОДЕ BCD С ПОВЫШЕННОЙ
                                ; ТОЧНОСТЬЮ. РЕЗУЛЬТАТ УМНОЖЕНИЯ
                                ; 1234 * 1234 = 1522756
                                ; В ПАМЯТИ AY1   = 56H
                                ;          AY1+1 = 27H
                                ;          AY1+2 = 52H
                                ;          AY1+3 = 01H
                                ;          AY1+4 = 00H
                                ;          AY1+5 = 00H
                                ;          AY1+6 = 00H

        JMP     SC6L

SZAYS   EQU     7               ;ДЛИНА МАССИВОВ В БАЙТАХ
AY1:
        DB      034H
```

```
        DB      012H
        DB      0
        DB      0
        DB      0
        DB      0
        DB      0

AY2:
        DB      034H
        DB      012H
        DB      0
        DB      0
        DB      0
        DB      0
        DB      0

        END
```

6М. ДЕСЯТИЧНОЕ ДЕЛЕНИЕ ЧИСЕЛ С ПОВЫШЕННОЙ ТОЧНОСТЬЮ (MPDDIV)

Делятся два многобайтных беззнаковых десятичных числа. Оба числа хранятся в памяти таким образом, что их самые младшие по значению байты занимают самые младшие адреса. Частное замещает делимое; остаток не возвращается, но его базовый адрес находится в ячейках памяти HDEPTR и HDEPTR + 1. Длина чисел – 255 байт или меньше. Если нет ошибок, то флаг переноса очищается; при попытке деления на 0 флаг переноса устанавливается в 1, делимое остается без изменения, а остаток равен 0.

Процедура. Деление осуществляется с помощью определения числа раз, которое делитель может быть вычтен из делимого. Это число сохраняется в частном, остаток делается новым делимым и циклически сдвигается влево на одну цифру делимое и частное. Если находится делитель, равный 0, то немедленно осуществляется выход из программы с установленным флагом переноса. В противном случае флаг переноса очищается. Вычитание осуществляется с помощью арифметических операций дополнения до десяти; для увеличения скорости результаты этих операций заменяются в делителе дополнением до девяти.

Используемые регистры: все.
Время выполнения: зависит от длины операндов и значения цифр в частном (определяющего число раз, которое делитель должен быть вычтен из делимого). Если в среднем цифры имеют значение 5, то время выполнения приблизительно равно $1128 * LENGTH^2 + 2722 * LENGTH + 393$ такта (8080) или $1150 * LENGTH^2 + 2647 * LENGTH + 354$ такта (8085), где LENGTH – число байтов в операндах.
Размер программы: 214 байт.
Память, необходимая для данных: 523 байта в любом месте ОЗУ. Это область для временного хранения старшей части делимого (255 байт, начиная с адреса HIDE1), результата вычитания (255 байт, начиная с адреса HIDE2), длины операндов (1 байт по адресу LENGTH), следующей цифры массива (1 байт по адресу NDIGIT), счетчика для цикла вычитаний (1 байт по адресу CNT), указателей для делимого, делителя, текущей старшей части делимого и остатка и другой старшей части делимого (по 2 байта на каждый, начиная с адресов DVADR, DSADR, HDEPTR и

ODEPTR, соответственно) и счетчика цикла деления (2 байта, начиная с адреса COUNT).

Специальные случаи:

1) длина 0 вызывает выход с установленным флагом переноса, частным, равным исходному делимому, и остатком, значение которого не определено,

2) делитель, равный 0, вызывает выход с флагом переноса, установленным в 1, частным, равным исходному делимому, и остатком, равным 0.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес делимого в регистрах H и L.
Базовый адрес делителя в регистрах D и E.
Длина операндов в байтах в регистрах B.

УСЛОВИЯ НА ВЫХОДЕ

Делимое заменяется делимым, деленным на делитель.
Если делитель ненулевой, то флаг переноса = 0 и результат нормальный.
Если делитель равен 0, то флаг переноса = 1, делимое остается без изменения, а остаток равен 0.
Базовый адрес остатка (т. е. адрес его младших по значению цифр) находится в HDEPTR и HDEPTR + 1.
Делитель заменяется его дополнением до девяти.

ПРИМЕР

1. Данные: длина операндов = 04 байта,
делимое = 22142298_{16},
делитель = 00006294_{16}.
Результат: делимое = 00003518_{16},
остаток (базовый адрес в HDEPTR и HDEPTR + 1) = 00000006_{16},
флаг переноса равен 0, что указывает на отсутствие ошибки деления на 0.

```
;                                                                  ;
;                                                                  ;
;                                                                  ;
;                                                                  ;
;       ЗАГОЛОВОК:     ДЕСЯТИЧНОЕ ДЕЛЕНИЕ ЧИСЕЛ С ПОВЫШЕННОЙ ТОЧНОСТЬЮ ;
;       ИМЯ:           MPDDIV                                      ;
;                                                                  ;
;                                                                  ;


;                                                                  ;
;       НАЗНАЧЕНИЕ:    ДЕЛИТ ДВА МАССИВА БАЙТ В КОДЕ BCD           ;
;                      ЧАСТНОЕ := ДЕЛИМОЕ / ДЕЛИТЕЛЬ               ;
;                                                                  ;
;       ВХОД:          РЕГИСТРЫ H И L = БАЗОВЫЙ АДРЕС ДЕЛИМОГО     ;
;                      РЕГИСТРЫ D И E = БАЗОВЫЙ АДРЕС ДЕЛИТЕЛЯ     ;
;                      РЕГИСТР B = ДЛИНА ОПЕРАНДОВ В БАЙТАХ        ;
;                                                                  ;
;                        КАЖДЫЙ МАССИВ ДОЛЖЕН СОДЕРЖАТЬ ОДНО ЧИСЛО В ;
;                        КОДЕ BCD БЕЗ ЗНАКА С МАКСИМАЛЬНОЙ ДЛИНОЙ  ;
;                        255 БАЙТ. ARRAY[0] ЯВЛЯЕТСЯ МЛАДШИМ БАЙТОМ, А ;
;                        ARRAY[LENGTH-1] - СТАРШИМ БАЙТОМ; ЗДЕСЬ   ;
;                        ARRAY - БАЗОВЫЙ АДРЕС МАССИВА, А LENGTH - ;
;                        ДЛИНА.                                    ;
```

```
;                                                                         ;
;       ВЫХОД:              ДЕЛИМОЕ := ДЕЛИМОЕ / ДЕЛИТЕЛЬ                 ;
;                           HDEPTR := БАЗОВЫЙ АДРЕС ОСТАТКА               ;
;                           ЕСЛИ НЕТ ОШИБОК, ТО                           ;
;                             ФЛАГ ПЕРЕНОСА := 0                          ;
;                           ИНАЧЕ                                         ;
;                             ОШИБКА ДЕЛЕНИЯ НА 0                         ;
;                             ФЛАГ ПЕРЕНОСА := 1                          ;
;                             ДЕЛИМОЕ ОСТАЕТСЯ БЕЗ ИЗМЕНЕНИЯ              ;
;                             ОСТАТОК := 0                                ;
;                                                                         ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ                                        ;
;                                                                         ;
;       ВРЕМЯ:              ЕСЛИ ПРИНЯТЬ, ЧТО СРЕДНЕЕ ЗНАЧЕНИЕ ЦИФРЫ      ;
;                           ЧАСТНОГО РАВНО 5,ТО ВРЕМЯ ПРИБЛИЗИТЕЛЬНО РАВНО ;
;                            (1128 * ДЛИНА^2) + (2722 * ДЛИНА) + 393 ТАКТА ;
;                            ДЛЯ 8080                                     ;
;                            (1150 * ДЛИНА^2) + (2647 * ДЛИНА) + 354 ТАКТА ;
;                            ДЛЯ 8085                                     ;
;                                                                         ;
;       РАЗМЕР:             ПРОГРАММА - 214 БАЙТ                          ;
;                           ДАННЫЕ    - 523 БАЙТА                         ;
;                                                                         ;
;                                                                         ;

MPDDIV:
        ;СОХРАНИТЬ ПАРАМЕТРЫ И ПРОВЕРИТЬ ДЛИНУ НА РАВЕНСТВО НУЛЮ
        SHLD    DVADR           ;СОХРАНИТЬ АДРЕС ДЕЛИМОГО
        XCHG
        SHLD    DSADR           ;СОХРАНИТЬ АДРЕС ДЕЛИТЕЛЯ
        MOV     A,B
        STA     LENGTH          ;СОХРАНИТЬ ДЛИНУ
        ORA     A               ;ПРОВЕРИТЬ ДЛИНУ
        JZ      OKEXIT          ;ВЫЙТИ ИЗ ПОДПРОГРАММЫ, ЕСЛИ ДЛИНА = 0

        ;ОБНУЛИТЬ ОБА БУФЕРА ДЕЛИМОГО И УСТАНОВИТЬ УКАЗАТЕЛИ ДЕЛИМОГО
        LXI     H,HIDE2
        SHLD    ODEPTR          ;ВТОРОЙ УКАЗАТЕЛЬ ДЕЛИМОГО = HIDE2
        XCHG                    ;DE = HIDE2
        LXI     H,HIDE1
        SHLD    HDEPTR          ;СТАРШИЙ УКАЗАТЕЛЬ ДЕЛИМОГО = HIDE1
        SUB     A               ;ВЗЯТЬ 0 ДЛЯ ЗАПОЛНЕНИЯ БУФЕРОВ
                                ;B = ДЛИНА БУФЕРОВ В БАЙТАХ

INITLP:
        MOV     M,A             ;ОБНУЛИТЬ БАЙТ HIDE1
        STAX    D               ;ОБНУЛИТЬ БАЙТ HIDE2
        INX     H               ;УВЕЛИЧИТЬ УКАЗАТЕЛИ ДЛЯ СЛЕДУЮЩЕГО
        INX     D               ; БАЙТА
        DCR     B
        JNZ     INITLP          ;ПРОДОЛЖАТЬ, ПОКА НЕ БУДЕТ ВЫПОЛНЕНО

        ;УСТАНОВИТЬ СЧЕТЧИК, РАВНЫЙ ЧИСЛУ ЦИФР + 1
        ; СЧЕТЧИК := (ДЛИНА * 2) + 1
        LDA     LENGTH
        MOV     L,A
        MVI     H,0
```

```
        DAD     H                   ;ДЛИНА * 2
        INX     H                   ;ДЛИНА * 2 + 1
        SHLD    COUNT               ;СЧЕТЧИК = ДЛИНА * 2 + 1

        ;ПРОВЕРИТЬ НА ДЕЛЕНИЕ НА НУЛЬ
        ;ВЫПОЛНИТЬ ЛОГИЧЕСКУЮ ОПЕРАЦИЮ "ИЛИ" ДЛЯ ВСЕГО ДЕЛИТЕЛЯ,
        ; ЧТОБЫ ПРОВЕРИТЬ, НЕ РАВНЫ ЛИ ВСЕ БАЙТЫ 0
        LHLD    DSADR               ;HL = АДРЕС ДЕЛИТЕЛЯ
        LDA     LENGTH
        MOV     B,A                 ;B = ДЛИНА МАССИВОВ В БАЙТАХ
        SUB     A                   ;НАЧАТЬ ВЫПОЛНЕНИЕ ЛОГИЧЕСКОЙ
                                    ; ОПЕРАЦИИ "ИЛИ" С 0

DV01:
        ORA     M                   ;ВЫПОЛНИТЬ ОПЕРАЦИЮ "ИЛИ" СЛЕДУЮЩЕГО
                                    ; БАЙТА ДЕЛИТЕЛЯ
        INX     H
        DCR     B
        JNZ     DV01
        ORA     A                   ;ПРОВЕРИТЬ ДЕЛИТЕЛЬ НА РАВЕНСТВО НУЛЮ
        JZ      EREXIT              ;ВЫХОД ПО ОШИБКЕ, ЕСЛИ ДЕЛИТЕЛЬ РАВЕН 0

        ;ВЗЯТЬ ДОПОЛНЕНИЕ ДЕЛИТЕЛЯ ДО 9, ТАК КАК 8080 ВЫПОЛНЯЕТ
        ; КОРРЕКЦИЮ В ДЕСЯТИЧНЫЙ ВИД ТОЛЬКО ПОСЛЕ СЛОЖЕНИЯ
        LHLD    DSADR               ;HL УКАЗЫВАЕТ НА ДЕЛИТЕЛЬ
        LDA     LENGTH
        MOV     B,A                 ;B = ДЛИНА В БАЙТАХ
D09S:
        MVI     A,99H               ;СФОРМИРОВАТЬ ДОПОЛНЕНИЕ БАЙТА ДО 9
        SUB     M
        MOV     M,A
        INX     H
        DCR     B
        JNZ     D09S                ;ПРОДОЛЖАТЬ ДЛЯ ВСЕХ БАЙТОВ

        SUB     A                   ;НАЧАТЬ СО СЛЕДУЮЩЕЙ ЦИФРЫ, РАВНОЙ 0
        STA     NDIGIT

        ;ВЫПОЛНИТЬ ДЕЛЕНИЕ С ПОМОЩЬЮ ПОВТОРЯЮЩИХСЯ ВЫЧИТАНИЙ

DVLOOP:
        ;СДВИНУТЬ ВЛЕВО ЦИКЛИЧЕСКИ МЛАДШУЮ ЧАСТЬ ДЕЛИМОГО И ЧАСТНОЕ:
        ; СТАРШАЯ ЦИФРА NDIGIT СТАНОВИТСЯ МЛАДШЕЙ ЦИФРОЙ ЧАСТНОГО
        ; (МАССИВ ДЕЛИМОГО), А СТАРШАЯ ЦИФРА МАССИВА ДЕЛИМОГО
        ; ПЕРЕХОДИТ В СТАРШУЮ ЦИФРУ NDIGIT
        LHLD    DVADR

        CALL    RLARY               ;СДВИНУТЬ ЦИКЛИЧЕСКИ МЛАДШУЮ ЧАСТЬ
                                    ; ДЕЛИМОГО

        ;ЕСЛИ СЧЕТЧИК = 0, ТО РАБОТА ВЫПОЛНЕНА
        LHLD    COUNT               ;УМЕНЬШИТЬ СЧЕТЧИК НА 1
        DCX     H
        SHLD    COUNT
        MOV     A,H                 ;ПРОВЕРИТЬ 16-РАЗРЯДНЫЙ СЧЕТЧИК
        ORA     L                   ; НА РАВЕНСТВО НУЛЮ
        JZ      OKEXIT              ;ВЫЙТИ, ЕСЛИ СЧЕТЧИК = 0
```

```
        ;СДВИНУТЬ ВЛЕВО ЦИКЛИЧЕСКИ СТАРШУЮ ЧАСТЬ ДЕЛИМОГО
        ; СТАРШАЯ ЦИФРА ДЕЛИМОГО СТАНОВИТСЯ СТАРШЕЙ ЦИФРОЙ NDIGIT
        LHLD    HDEPTR
        CALL    RLARY               ;СДВИНУТЬ ЦИКЛИЧЕСКИ СТАРШУЮ ЧАСТЬ
                                    ; ДЕЛИМОГО

        ;ПОСМОТРЕТЬ, СКОЛЬКО РАЗ ДЕЛИТЕЛЬ ПЕРЕХОДИТ В СТАРШУЮ ЧАСТЬ
        ; ДЕЛИМОГО; НА ВЫХОДЕ ИЗ ЭТОГО ЦИКЛА СТАРШАЯ ЦИФРА NDIGIT
        ; ЯВЛЯЕТСЯ СЛЕДУЮЩЕЙ ЦИФРОЙ ЧАСТНОГО, А СТАРШАЯ ЧАСТЬ ДЕЛИМОГО -
        ; ОСТАТКОМ
        SUB     A                   ;ДЛЯ НАЧАЛА ОЧИСТИТЬ СЧЕТЧИК
        STA     NDIGIT


SUBLP:
        LHLD    HDEPTR
        XCHG                        ;DE УКАЗЫВАЕТ НА СТАРШУЮ ЧАСТЬ ДЕЛИМОГО
        LHLD    ODEPTR
        MOV     C,L
        MOV     B,H                 ;BC = ДРУГАЯ СТАРШАЯ ЧАСТЬ ДЕЛИМОГО
        LHLD    DSADR               ;HL = ДОПОЛНЕНИЕ ДЕЛИТЕЛЯ ДО ДЕВЯТИ
        LDA     LENGTH              ;СЧЕТЧИК = ДЛИНА В БАЙТАХ
        STA     CNT
        STC                         ;УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА ДЛЯ ДОПОЛНЕНИЯ
                                    ; ДО 10

        ;ВЫЧЕСТЬ ДЕЛИТЕЛЬ ИЗ ДЕЛИМОГО, ИСПОЛЬЗУЯ ПРИБАВЛЕНИЕ ДОПОЛНЕНИЯ
        ; ДО 10 (ДОПОЛНЕНИЯ ДО 9 ПЛЮС 1)
        ;В КОНЦЕ ФЛАГ ПЕРЕНОСА ЯВЛЯЕТСЯ ИНВЕРТИРОВАННЫМ ЗАЕМОМ
INNER:
        LDAX    D                   ;ВЗЯТЬ СЛЕДУЮЩИЙ БАЙТ ДЕЛИМОГО
        ADC     M                   ;ПРИБАВИТЬ ДОПОЛНЕНИЕ ДЕЛИТЕЛЯ ДО 10
        DAA                         ;ПРЕОБРАЗОВАТЬ В ДЕСЯТИЧНЫЙ ВИД
        STAX    B                   ;ЗАПОМНИТЬ РЕЗУЛЬТАТ ВО ВТОРОМ ДЕЛИМОМ
        INX     H                   ;УВЕЛИЧИТЬ УКАЗАТЕЛИ ДЛЯ СЛЕДУЮЩЕГО
        INX     D                   ; БАЙТА
        INX     B
        LDA     CNT                 ;УМЕНЬШИТЬ СЧЕТЧИК НА 1
        DCR     A
        STA     CNT
        JNZ     INNER               ;ПРОДОЛЖИТЬ ДЛЯ ВСЕХ БАЙТОВ
                                    ; ФЛАГ ПЕРЕНОСА ЯВЛЯЕТСЯ
                                    ; ИНВЕРТИРОВАННЫМ ЗАЕМОМ
        JNC     DVLOOP              ;ПЕРЕЙТИ,КОГДА ПРОИСХОДИТ ЗАЕМ
                                    ;ПРИ ЭТОМ NDIGIT СОДЕРЖИТ ЧИСЛО
                                    ; ПЕРЕХОДОВ ДЕЛИТЕЛЯ В НАЧАЛЬНУЮ
                                    ; СТАРШУЮ ЧАСТЬ ДЕЛИМОГО, А СТАРШАЯ
                                    ; ЧАСТЬ ДЕЛИМОГО СОДЕРЖИТ ОСТАТОК

        ;РАЗНОСТЬ НЕ ОТРИЦАТЕЛЬНА, ПОЭТОМУ ПРИБАВИТЬ 1 К СТАРШЕЙ ЦИФРЕ
        ; ЧИСЛА УСПЕШНЫХ ВЫЧИТАНИЙ NDIGIT
        LDA     NDIGIT              ;NDIGIT = NDIGIT + ШЕСТНАДЦАТЕРИЧНОЕ
        ADI     10H                 ; ЧИСЛО 10
        STA     NDIGIT
```

```
        ;ПОМЕНЯТЬ УКАЗАТЕЛИ, СДЕЛАВ ТАКИМ ОБРАЗОМ РАЗНОСТЬ НОВЫМ ДЕЛИМЫМ
        LHLD    HDEPTR
        XCHG
        LHLD    ODEPTR
        SHLD    HDEPTR
        XCHG
        SHLD    ODEPTR
        JMP     SUBLP                   ;ПРОДОЛЖАТЬ, ПОКА РАЗНОСТЬ НЕ СТАНЕТ
                                        ; ОТРИЦАТЕЛЬНОЙ

        ;НЕТ ОШИБОК, ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА
OKEXIT:
        ORA     A                       ;ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА, РЕЗУЛЬТАТ
                                        ; ПРАВИЛЬНЫЙ
        RET

        ;ОШИБКА ДЕЛЕНИЯ НА НУЛЬ, УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА
EREXIT:
        STC                             ;УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА, РЕЗУЛЬТАТ
                                        ; НЕПРАВИЛЬНЫЙ
        RET

        ;*********************************
        ;ПОДПРОГРАММА: RLARY
        ;НАЗНАЧЕНИЕ:   СДВИГАЕТ МАССИВ ЦИКЛИЧЕСКИ ВЛЕВО НА ОДНУ
        ;              ЦИФРУ (4 РАЗРЯДА)
        ;ВХОД:  HL = БАЗОВЫЙ АДРЕС МАССИВА
        ;       СДВИГ ПРОИЗВОДИТСЯ ЧЕРЕЗ СТАРШУЮ ЦИФРУ NDIGIT
        ;ВЫХОД: МАССИВ СДВИГАЕТСЯ ЦИКЛИЧЕСКИ ВЛЕВО
        ;       ЧЕРЕЗ СТАРШУЮ ЦИФРУ NDIGIT
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ
        ;*********************************

RLARY:
        ;СДВИНУТЬ NDIGIT В МЛАДШУЮ ЦИФРУ МАССИВА
        ; И СДВИНУТЬ МАССИВ ВЛЕВО
        LDA     LENGTH
        MOV     B,A                     ;B = ДЛИНА В БАЙТАХ
        LDA     NDIGIT
        MOV     E,A                     ;E = NDIGIT
                                        ;HL = БАЗОВЫЙ АДРЕС МАССИВА

SHIFT:
        MOV     A,M                     ;ВЗЯТЬ СЛЕДУЮЩИЙ БАЙТ
        MOV     D,A                     ;СОХРАНИТЬ В D
        ANI     0FH                     ;ОЧИСТИТЬ СТАРШУЮ ЦИФРУ
        ORA     E                       ;СГРУППИРОВАТЬ NDIGIT И БАЙТ
        RRC                             ;ПЕРЕСЛАТЬ МЛАДШУЮ ЦИФРУ
        RRC                             ; В СТАРШУЮ ЦИФРУ
        RRC
        RRC
        MOV     M,A                     ;ЗАПОМНИТЬ МЛАДШУЮ ЦИФРУ В МАССИВЕ
        MOV     A,D
        ANI     0F0H                    ;ОЧИСТИТЬ МЛАДШУЮ ЦИФРУ
        MOV     E,A                     ;СОХРАНИТЬ СЛЕДУЮЩЕЕ ЗНАЧЕНИЕ NDIGIT
                                        ; В РЕГИСТРЕ E
        INX     H                       ;УВЕЛИЧИТЬ УКАЗАТЕЛЬ ДЛЯ СЛЕДУЮЩЕГО БАЙТА
        DCR     B                       ;УМЕНЬШИТЬ СЧЕТЧИК
```

```
        JNZ     SHIFT                   ;ПРОДОЛЖАТЬ, ПОКА НЕ БУДУТ СДВИНУТЫ
                                        ; ВСЕ БАЙТЫ
        MOV     A,E
        STA     NDIGIT                  ;ЗАПОМНИТЬ NDIGIT
        RET

        ;ДАННЫЕ
LENGTH: DS      1                       ;ДЛИНА МАССИВОВ В БАЙТАХ
NDIGIT: DS      1                       ;СЛЕДУЮЩАЯ ЦИФРА В МАССИВЕ
CNT:    DS      1                       ;СЧЕТЧИК ДЛЯ ЦИКЛА ВЫЧИТАНИЯ
DVADR:  DS      2                       ;АДРЕС ДЕЛИМОГО
DSADR:  DS      2                       ;АДРЕС ДЕЛИТЕЛЯ
HDEPTR: DS      2                       ;УКАЗАТЕЛЬ СТАРШЕЙ ЧАСТИ ДЕЛИМОГО
ODEPTR: DS      2                       ;УКАЗАТЕЛЬ ДРУГОЙ ЧАСТИ ДЕЛИМОГО
COUNT:  DS      2                       ;СЧЕТЧИК ЦИКЛА ДЕЛЕНИЯ
HIDE1:  DS      255                     ;БУФЕР 1 СТАРШЕЙ ЧАСТИ ДЕЛИМОГО
HIDE2:  DS      255                     ;БУФЕР 2 СТАРШЕЙ ЧАСТИ ДЕЛИМОГО
;                                                                       ;
;                                                                       ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                               ;
;                                                                       ;
;                                                                       ;

SC6H:
        LXI     H,AY1                   ;БАЗОВЫЙ АДРЕС ДЕЛИМОГО
        LXI     D,AY2                   ;БАЗОВЫЙ АДРЕС ДЕЛИТЕЛЯ
        MVI     B,SZAYS                 ;ДЛИНА МАССИВОВ В БАЙТАХ
        CALL    MPDDIV                  ;ДЕЛЕНИЕ ЧИСЕЛ В КОДЕ BCD С ПОВЫШЕННОЙ
                                        ; ТОЧНОСТЬЮ. РЕЗУЛЬТАТ ДЕЛЕНИЯ
                                        ; 1522756 / 1234 = 1234
                                        ; В ПАМЯТИ AY1   = 34H
                                        ;          AY1+1 = 12H
                                        ;          AY1+2 = 00H
                                        ;          AY1+3 = 00H
                                        ;          AY1+4 = 00H
                                        ;          AY1+5 = 00H
                                        ;          AY1+6 = 00H

        JMP     SC6H

SZAYS   EQU     7                       ;ДЛИНА МАССИВОВ В БАЙТАХ
AY1:
        DB      056H
        DB      027H
        DB      052H
        DB      01H
        DB      0
        DB      0
        DB      0

AY2:
        DB      034H
        DB      012H
        DB      0
        DB      0
        DB      0
        DB      0
        DB      0

        END
```

6N. ДЕСЯТИЧНОЕ СРАВНЕНИЕ ЧИСЕЛ С ПОВЫШЕННОЙ ТОЧНОСТЬЮ

Сравниваются два многобайтных беззнаковых десятичных (BCD) числа и соответствующим образом устанавливаются флаги переноса и нуля. Флаг нуля равен 1, если операнды равны, и 0, если они не равны. Флаг переноса устанавливается в 1, если вычитаемое больше уменьшаемого; в противном случае флаг переноса очищается. Таким образом, флаги устанавливаются так, как если бы вычитаемое вычиталось из уменьшаемого.

Примечание. Эта программа в точности такая же, как подпрограмма 6I двоичного сравнения с повышенной точностью, поскольку если операнды только сравниваются, то их форма не имеет никакого значения. Распечатку и другие подробности см. в подпрограмме 6I.

ПРИМЕРЫ

1. Данные: длина операндов = 6 байт,
   вычитаемое = 196528719340_{16},
   уменьшаемое = 456780153266_{16}.
   Результат: флаг нуля = 0 (операнды не равны),
   флаг переноса = 0 (вычитаемое не больше уменьшаемого).
2. Данные: длина операндов = 6 байт,
   вычитаемое = 196528719340_{16},
   уменьшаемое = 196528719340_{16}.
   Результат: флаг нуля равен 1 (операнды равны),
   флаг переноса = 0 (вычитаемое не больше уменьшаемого).
3. Данные: длина операнда = 6 байт,
   вычитаемое = 196528719340_{16},
   уменьшаемое = 073785991074_{16}.
   Результат: флаг нуля = 0 (операнды не равны),
   флаг переноса = 1 (вычитаемое больше уменьшаемого).

ГЛАВА 7

РАБОТА С РАЗРЯДАМИ И СДВИГИ

7A. УСТАНОВКА РАЗРЯДА (BITSET)

Указанный разряд байта устанавливается в 1.

Процедура. Выполняется логическая операция ИЛИ заданного байта с маской, содержащей 1 в выбранном разряде и нули в остальных. Маски, имеющие по одному единичному разряду, находятся в таблице.

Используемые регистры: AF, BC, HL.
Время выполнения: 61 такт (8080) или 59 тактов (8085).
Размер программы: 20 байт.
Память, необходимая для данных: отсутствует.
Специальный случай: номер разряда выше 7-го интерпретируется по модулю 8 (разряд номер 9 эквивалентен разряду номер 1).

УСЛОВИЯ НА ВХОДЕ

Номер разряда, который должен быть установлен, в регистре А.
Байт данных в регистре В.

УСЛОВИЯ НА ВЫХОДЕ

Результат (байт с установленным разрядом) в регистре А.

ПРИМЕРЫ

1. Данные: (B) = $6E_{16}$ = 01101110_2 (данные),
   (A) = 4 (номер разряда, который должен быть установлен).
   Результат: (A) = $7E_{16}$ = 01111110_2 (данные с разрядом 4, установленным в 1).
2. Данные: (B) = 39_{16} = 00111001_2 (данные),
   (A) = 2 (номер разряда, который должен быть установлен).
   Результат: (A) = $3D_{16}$ = 00111101_2 (данные с разрядом 2, установленным в 1).

```
;                                                                   ;
;                                                                   ;
;                                                                   ;
;                                                                   ;
;       ЗАГОЛОВОК:      УСТАНОВКА РАЗРЯДА                           ;
;       ИМЯ:            BITSET                                      ;
;                                                                   ;
;                                                                   ;


;                                                                   ;
;       НАЗНАЧЕНИЕ:     УСТАНАВЛИВАЕТ РАЗРЯД В БАЙТЕ                ;
;                                                                   ;
;       ВХОД:           РЕГИСТР A = НОМЕР РАЗРЯДА, КОТОРЫЙ ДОЛЖЕН БЫТЬ ;
;                                   УСТАНОВЛЕН                      ;
;                       РЕГИСТР B = БАЙТ С ДАННЫМИ                  ;
;                                                                   ;
;       ВЫХОД:          РЕГИСТР A = БАЙТ ДАННЫХ С УСТАНОВЛЕННЫМ РАЗРЯДОМ;
;                                                                   ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF,BC,HL                             ;
;                                                                   ;
;       ВРЕМЯ:          59 ТАКТОВ ДЛЯ 8085, 61 ТАКТ ДЛЯ 8080        ;
;                                                                   ;
;       РАЗМЕР:         ПРОГРАММА - 20 БАЙТ                         ;
;                                                                   ;
;                                                                   ;

        ;УСТАНОВИТЬ РАЗРЯД С ПОМОЩЬЮ ЛОГИЧЕСКОЙ ОПЕРАЦИИ "ИЛИ"
        ; С 1 В СООТВЕТСТВУЮЩЕЙ ПОЗИЦИИ
        ;ПОЛУЧИТЬ МАСКУ, ИСПОЛЬЗУЯ НОМЕР РАЗРЯДА В КАЧЕСТВЕ ИНДЕКСА
        ; В МАССИВЕ BITMSK
BITSET:
        ANI     00000111B       ;ОГРАНИЧИТЬ ЧИСЛО РАЗРЯДОВ 0...7
        MOV     C,A
        MOV     A,B
        LXI     H,BITMSK        ;БАЗОВЫЙ АДРЕС МАССИВА BITMSK
        MVI     B,0             ;РАСШИРИТЬ НОМЕР РАЗРЯДА ДО 16 РАЗРЯДОВ
        DAD     B               ;ПОЛУЧИТЬ АДРЕС В МАССИВЕ BITMSK
        ORA     M               ;УСТАНОВИТЬ РАЗРЯД
        RET
```

```
        ;ТАБЛИЦА МАСОК С ОДНИМ УСТАНОВЛЕННЫМ РАЗРЯДОМ
BITMSK: DB      00000001B       ;РАЗРЯД 0 = 1
        DB      00000010B       ;РАЗРЯД 1 = 1
        DB      00000100B       ;РАЗРЯД 2 = 1
        DB      00001000B       ;РАЗРЯД 3 = 1
        DB      00010000B       ;РАЗРЯД 4 = 1
        DB      00100000B       ;РАЗРЯД 5 = 1
        DB      01000000B       ;РАЗРЯД 6 = 1
        DB      10000000B       ;РАЗРЯД 7 = 1


;                                                       ;
;                                                       ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                               ;
;                                                       ;
;                                                       ;

SC7A:
        MVI     B,0             ;РЕГИСТР B = ДАННЫЕ
        MVI     A,3             ;РЕГИСТР A = НОМЕР РАЗРЯДА (0...7)
        CALL    BITSET          ;РЕЗУЛЬТАТ = ДАННЫЕ С УСТАНОВЛЕННЫМ
                                ;  РАЗРЯДОМ 3 = 00001000B (08H)

        JMP     SC7A

        END
```

7B. ОЧИСТКА РАЗРЯДА (BITCLR)

Очищается заданный разряд в байте.

Процедура. Выполняется логическая операция И данных с маской, содержащей 0 в выбранном разряде и нули в остальных. Маски, имеющие по одному нулевому разряду, находятся в таблице.

> **Используемые регистры:** AF, BC, HL.
> **Время выполнения:** 61 такт (8080) или 59 тактов (8085).
> **Размер программы:** 20 байт.
> **Память, необходимая для данных:** отсутствует.
> **Специальный случай:** номер разряда свыше 7 интерпретируется по модулю 8 (разряд номер 12 эквивалентен разряду номер 3).

УСЛОВИЯ НА ВХОДЕ

Номер разряда, который должен быть очищен, в регистре A.
Байт данных в B.

УСЛОВИЯ НА ВЫХОДЕ

Результат (данные с очищенным разрядом) в регистре A.

ПРИМЕРЫ

1. Данные: (B) = $6E_{16} \doteq 01101110_2$ (данные),
   (A) = 6 (номер разряда, который должен быть очищен).
   Результат: (A) = $2E_{16} = 00101110_2$ (данные с очищенным разрядом 6).

2\. Данные: (B) = $39_{16} = 00111001_2$ (данные).
(A) = 4 (номер разряда, который должен быть очищен).
Результат: (A) = $29_{16} = 00101001_2$ (данные с очищенным разрядом 4).

```
;                                                                        ;
;                                                                        ;
;                                                                        ;
;                                                                        ;
;       ЗАГОЛОВОК:      ОЧИСТКА РАЗРЯДА                                  ;
;       ИМЯ:            BITCLR                                           ;
;                                                                        ;
;                                                                        ;

;                                                                        ;
;       НАЗНАЧЕНИЕ:     ОЧИЩАЕТ РАЗРЯД В БАЙТЕ                           ;
;                                                                        ;
;       ВХОД:           РЕГИСТР A = НОМЕР РАЗРЯДА, КОТОРЫЙ ДОЛЖЕН БЫТЬ   ;
;                                   ОЧИЩЕН                               ;
;                       РЕГИСТР B = БАЙТ С ДАННЫМИ                       ;
;                                                                        ;
;       ВЫХОД:          РЕГИСТР A = БАЙТ ДАННЫХ С ОЧИЩЕННЫМ РАЗРЯДОМ     ;
;                                                                        ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF,BC,HL                                  ;
;                                                                        ;
;       ВРЕМЯ:          59 ТАКТОВ ДЛЯ 8085, 61 ТАКТ ДЛЯ 8080             ;
;                                                                        ;
;       РАЗМЕР:         ПРОГРАММА - 20 БАЙТ                              ;
;                                                                        ;
;                                                                        ;

BITCLR:
        ANI     00000111B       ;ОГРАНИЧИТЬ ЧИСЛО РАЗРЯДОВ 0...7
        MOV     C,A
        MOV     A,B
        LXI     H,BITMSK        ;БАЗОВЫЙ АДРЕС МАССИВА BITMSK
        MVI     B,0             ;РАСШИРИТЬ НОМЕР РАЗРЯДА ДО 16 РАЗРЯДОВ
        DAD     B               ;ПОЛУЧИТЬ АДРЕС В МАССИВЕ BITMSK
        ANA     M               ;ОЧИСТИТЬ РАЗРЯД
        RET

        ;ТАБЛИЦА МАСОК С ОДНИМ ОЧИЩЕННЫМ РАЗРЯДОМ
BITMSK: DB      11111110B       ;РАЗРЯД 0 = 0
        DB      11111101B       ;РАЗРЯД 1 = 0
        DB      11111011B       ;РАЗРЯД 2 = 0
        DB      11110111B       ;РАЗРЯД 3 = 0
        DB      11101111B       ;РАЗРЯД 4 = 0
        DB      11011111B       ;РАЗРЯД 5 = 0
        DB      10111111B       ;РАЗРЯД 6 = 0
        DB      01111111B       ;РАЗРЯД 7 = 0


;                                                                        ;
;                                                                        ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                                ;
;                                                                        ;
;                                                                        ;
```

```
SC7B:
        MVI     B,11111111B     ;РЕГИСТР B = ДАННЫЕ
        MVI     A,3             ;РЕГИСТР A = НОМЕР РАЗРЯДА (0...7)
        CALL    BITCLR          ;РЕЗУЛЬТАТ = ДАННЫЕ С ОЧИЩЕННЫМ
                                ;  РАЗРЯДОМ 3 = 11110111B = 0F7H

        JMP     SC7B

        END
```

7С. ПРОВЕРКА РАЗРЯДА (BITTST)

Значение флага нуля устанавливается обратным значению выбранного разряда в байте.

Процедура. Выполняется логическая операция И данных с маской, содержащей 1 в выбранном разряде и нули в остальных. Результат равен нулю, если проверяемый разряд содержит 0, и не равен нулю, если проверяемый разряд содержит 1. Следовательно, значение флага нуля устанавливается обратным значению проверяемого разряда.

Используемые регистры: AF, BC, HL.
Время выполнения: 61 такт (8080) или 59 тактов (8085).
Размер программы: 20 байт.
Память, необходиман для данных: отсутствует.
Специальный случай: номер разряда свыше 7 интерпретируется по модулю 8 (номер разряда 10 эквивалентен номеру разряда 2).

УСЛОВИЯ НА ВХОДЕ

Номер проверяемого разряда в регистре A.
Байт данных в регистре B.

УСЛОВИЯ НА ВЫХОДЕ

Значение флага нуля устанавливается обратным значению заданного разряда в байте данных. (Флаг нуля установлен, если разряд равен 0, и очищен, если разряд равен 1.)

ПРИМЕРЫ

1. Данные: (B) = $6E_{16}$ = 01101110_2 (данные),
   (A) = 3 (номер проверяемого разряда).
   Результат: флаг нуля = 0 (значение, обратное разряду 3 данных).
2. Данные: (B) = 39_{16} = 00111001_2 (данные),
   (A) = 6 (номер проверяемого разряда).
   Результат: флаг нуля = 1 (значение, обратное разряду 6 данных).

```
;                                                                   ;
;                                                                   ;
;                                                                   ;
;                                                                   ;
;       ЗАГОЛОВОК:      ПРОВЕРКА РАЗРЯДА                            ;
;       ИМЯ:            BITTST                                      ;
;                                                                   ;
;                                                                   ;
```

```
;                                                                            ;
;       НАЗНАЧЕНИЕ:        ПРОВЕРЯЕТ РАЗРЯД В БАЙТЕ                          ;
;                                                                            ;
;       ВХОД:              РЕГИСТР A = НОМЕР РАЗРЯДА, КОТОРЫЙ ДОЛЖЕН БЫТЬ    ;
;                                      ПРОВЕРЕН                              ;
;                          РЕГИСТР B = БАЙТ С ДАННЫМИ                        ;
;                                                                            ;
;       ВЫХОД:             Z = 1, ЕСЛИ РАЗРЯД РАВЕН 0                        ;
;                          Z = 0, ЕСЛИ РАЗРЯД РАВЕН 1                        ;
;                                                                            ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF,BC,HL                                      ;
;                                                                            ;
;       ВРЕМЯ:             59 ТАКТОВ ДЛЯ 8085, 61 ТАКТ ДЛЯ 8080              ;
;                                                                            ;
;       РАЗМЕР:            ПРОГРАММА - 20 БАЙТ                               ;
;                                                                            ;
;                                                                            ;


BITTST:
        ANI     00000111B       ;ОГРАНИЧИТЬ ЧИСЛО РАЗРЯДОВ ДО 0...7
        MOV     C,A
        MOV     A,B
        LXI     H,BITMSK        ;БАЗОВЫЙ АДРЕС МАССИВА BITMSK
        MVI     B,0             ;РАСШИРИТЬ НОМЕР РАЗРЯДА ДО 16 РАЗРЯДОВ
        DAD     B               ;ПОЛУЧИТЬ АДРЕС В МАССИВЕ BITMSK
        ANA     M               ;ПРОВЕРИТЬ РАЗРЯД
        RET

        ;ТАБЛИЦА МАСОК С ОДНИМ УСТАНОВЛЕННЫМ РАЗРЯДОМ
BITMSK: DB      00000001B       ;РАЗРЯД 0 = 1
        DB      00000010B       ;РАЗРЯД 1 = 1
        DB      00000100B       ;РАЗРЯД 2 = 1
        DB      00001000B       ;РАЗРЯД 3 = 1
        DB      00010000B       ;РАЗРЯД 4 = 1
        DB      00100000B       ;РАЗРЯД 5 = 1
        DB      01000000B       ;РАЗРЯД 6 = 1
        DB      10000000B       ;РАЗРЯД 7 = 1



;                                                                            ;
;                                                                            ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                                    ;
;                                                                            ;
;                                                                            ;


SC7C:
        MVI     B,11110111B     ;РЕГИСТР B = БАЙТ ДАННЫХ
        MVI     A,3             ;РЕГИСТР A = НОМЕР РАЗРЯДА (0...7)
        CALL    BITTST          ;Z = 1, ТАК КАК РАЗРЯД 3 ДАННЫХ = 0

        JMP     SC7C

        END
```

Из байта выделяется поле разрядов, и это поле возвращается в младших по значению разрядах. Параметрами являются ширина поля и номер его младшего разряда.

Процедура. Маска с заданным числом единичных разрядов получается из таблицы, сдвигается влево для выравнивания ее с заданным младшим разрядом поля и с помощью логической операции И маски и поля получается поле. Затем поле разрядов нормируется с помощью сдвига его вправо таким образом, чтобы оно начиналось с разряда 0.

Используемые регистры: все.

Время выполнения: 23 * ПОЗИЦИЯ МЛАДШЕГО РАЗРЯДА плюс 112 тактов (8080) или 22 * ПОЗИЦИЯ МЛАДШЕГО РАЗРЯДА плюс 100 тактов (8085). (Позиция младшего разряда определяет, сколько раз маска должна быть сдвинута влево, а поле разрядов – вправо.)

Размер программы: 38 байт.

Память, необходимая для данных: отсутствует.

Специальные случаи:

1. Запрос поля, которое выходит за конец байта, вызывает выход программы с полем разрядов вплоть до 7-го. Таким образом, не происходит циклического перехода на начало байта. Если, например, пользователь запрашивает 6-разрядное поле, начинающееся с разряда 5, то программа возвращает только 3 разряда (с 5-го по 7-й).

2. Как позиция младшего разряда, так и число разрядов в поле интерпретируются по модулю 8. Таким образом, например, позиция разряда 11 эквивалентна позиции разряда 3, а поле из 10 разрядов эквивалентно полю из 2 разрядов. Заметим, однако, что число разрядов в поле интерпретируется в диапазоне от 1 до 8, т. е. поле из 16 разрядов эквивалентно полю из 8 разрядов, а не полю из 0 разрядов.

3. Запрос поля с числом разрядов 0 вызывает выход с результатом, равным 0.

УСЛОВИЯ НА ВХОДЕ

Позиция начального (самого младшего) разряда в поле (от 0 до 7) в регистре A.

Число разрядов в поле (от 1 до 8) в регистре D.

Байт данных в регистре E.

УСЛОВИЯ НА ВЫХОДЕ

Поле разрядов в регистре A (нормированное к разряду 0).

ПРИМЕРЫ

1. Данные: значение данных = $F6_{16} = 11110110_2$,
позиция самого младшего разряда = 4,
число разрядов в поле = 3.
Результат: поле разрядов = $07_{16} = 00000111_2$,
было выделено три разряда, начиная с разряда 4 (т. е. разряды с 4 по 6).

2. Данные: значение данных = $A2_{16} = 10100010_2$,
позиция самого младшего разряда = 6,
число разрядов в поле = 5.
Результат: поле разрядов = $02_{16} = 00000010_2$,
было выделено два разряда, начиная с разряда 6 (т. е. разряды 6 и 7); это было все, что имелось в наличии, хотя и было запрошено 5 разрядов.

```
;                                                                        ;
;                                                                        ;
;                                                                        ;
;                                                                        ;
;       ЗАГОЛОВОК:      ВЫДЕЛЕНИЕ ПОЛЯ РАЗРЯДОВ                          ;
;       ИМЯ:            BFE                                              ;
;                                                                        ;
;                                                                        ;


;                                                                        ;
;       НАЗНАЧЕНИЕ:     ВЫДЕЛЯЕТ ПОЛЕ РАЗРЯДОВ ИЗ БАЙТА И ВОЗВРАЩАЕТ     ;
;                       ЭТО ПОЛЕ В ВИДЕ, НОРМИРОВАННОМ К РАЗРЯДУ 0       ;
;                       ЗАМЕЧАНИЕ: ЕСЛИ ТРЕБУЕМОЕ ПОЛЕ СЛИШКОМ ВЕЛИКО,   ;
;                                  ТО БУДУТ ВОЗВРАЩЕНЫ ТОЛЬКО РАЗРЯДЫ    ;
;                                  ДО 7-ГО ВКЛЮЧИТЕЛЬНО. НАПРИМЕР, ЕСЛИ  ;
;                                  ТРЕБУЕТСЯ ВЫДЕЛИТЬ 4-РАЗРЯДНОЕ        ;
;                                  ПОЛЕ, НАЧИНАЯ С РАЗРЯДА 7, ТО БУДЕТ   ;
;                                  ВОЗВРАЩЕН ТОЛЬКО ОДИН РАЗРЯД (РАЗРЯД  ;
;                                  7).                                   ;
;                                                                        ;
;       ВХОД:           РЕГИСТР A = НОМЕР НАЧАЛЬНОГО (МЛАДШЕГО) РАЗРЯДА  ;
;                                   В ПОЛЕ (0...7)                       ;
;                       РЕГИСТР D = ЧИСЛО РАЗРЯДОВ В ПОЛЕ (1...8)        ;
;                       РЕГИСТР E = БАЙТ ДАННЫХ                          ;
;                                                                        ;
;       ВЫХОД:          РЕГИСТР A = ПОЛЕ                                 ;
;                                                                        ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ                                       ;
;                                                                        ;
;       ВРЕМЯ:          112 ТАКТОВ ПЛЮС (23 * ПОЗИЦИЯ МЛАДШЕГО РАЗРЯДА)  ;
;                        ТАКТОВ ДЛЯ 8080                                 ;
;       ВРЕМЯ:          100 ТАКТОВ ПЛЮС (22 * ПОЗИЦИЯ МЛАДШЕГО РАЗРЯДА)  ;
;                        ТАКТОВ ДЛЯ 8085                                 ;
;                                                                        ;
;       РАЗМЕР:         ПРОГРАММА - 38 БАЙТ                              ;
;                                                                        ;
;                                                                        ;

BFE:
        ;СДВИНУТЬ ДАННЫЕ ДЛЯ ТОГО, ЧТОБЫ НОРМИРОВАТЬ К РАЗРЯДУ 0
        ; ЕСЛИ РАЗРЯД В МЛАДШЕЙ ПОЗИЦИИ УЖЕ РАВЕН 0, ТО СДВИГ НЕ НУЖЕН
        ANI     00000111B       ;ОГРАНИЧИТЬ МЛАДШИЙ РАЗРЯД ДО 0...7
        JZ      EXTR            ;ПЕРЕЙТИ, ЕСЛИ МЛАДШИЙ РАЗРЯД РАВЕН 0,
                                ; НЕ СДВИГАТЬ
        MOV     C,A             ;ЧИСЛО СДВИГОВ = ПОЗИЦИЯ
                                ; МЛАДШЕГО РАЗРЯДА В ПОЛЕ
        MOV     A,E

SHFT:
        ORA     A               ;ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА
        RAR                     ;СДВИНУТЬ ДАННЫЕ ЛОГИЧЕСКИ ВПРАВО
        DCR     C
        JNZ     SHFT            ;ПРОДОЛЖИТЬ ДО НОРМИРОВАНИЯ
        MOV     E,A             ;СОХРАНИТЬ НОРМИРОВАННЫЕ ДАННЫЕ

        ;ВЫДЕЛИТЬ ПОЛЕ С ПОМОЩЬЮ МАСКИ С ЕДИНИЦАМИ
EXTR:
        MOV     A,D             ;ШИРИНА ПОЛЯ 0?
```

```
        ORA     A
        RZ                      ;ДА, ВЫЙТИ С ПОЛЕМ = 0
        DCR     A               ;ИНДЕКС = ШИРИНА - 1
        ANI     00000111B       ;ДОПУСКАЕТСЯ ТОЛЬКО ОТ 0 ДО 7
        MOV     C,A             ;BC = ИНДЕКС В МАССИВЕ МАСОК
        MVI     B,0
        LXI     H,MSKARY        ;HL = БАЗОВЫЙ АДРЕС В МАССИВЕ МАСОК
        DAD     B
        MOV     A,E             ;ВЗЯТЬ ДАННЫЕ
        ANA     M               ;ЗАМАСКИРОВАТЬ РАЗРЯДЫ, КОТОРЫЕ НЕ НУЖНЫ
        RET

        ;МАССИВ МАСОК, ИМЕЮЩИЙ ОТ 1 ДО 8 ЕДИНИЧНЫХ РАЗРЯДОВ
MSKARY: DB      00000001B
        DB      00000011B
        DB      00000111B
        DB      00001111B
        DB      00011111B
        DB      00111111B
        DB      01111111B
        DB      11111111B


;                                                                       ;
;                                                                       ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                               ;
;                                                                       ;
;                                                                       ;

SC7D:
        MVI     E,00011000B     ;РЕГИСТР E = БАЙТ ДАННЫХ
        MVI     D,3             ;РЕГИСТР D = ЧИСЛО РАЗРЯДОВ
        MVI     A,2             ;АККУМУЛЯТОР = МЛАДШИЙ РАЗРЯД
        CALL    BFE             ;ВЫДЕЛИТЬ ТРИ РАЗРЯДА, НАЧИНАЯ СО 2-ГО

        JMP     SC7D            ; РЕЗУЛЬТАТ = 00000110B

        END
```

7E. ЗАПИСЬ ПОЛЯ РАЗРЯДОВ (BFI)

Поле разрядов вставляется в байт. Параметрами являются ширина поля и позиция начального (самого младшего) разряда.

Процедура. Маска с заданным числом нулевых разрядов получается из таблицы. Затем маска и поле разрядов сдвигаются влево для выравнивания их с заданной позицией самого младшего разряда. Выполняется логическая операция И маски с исходным байтом данных и, таким образом, очищаются заданные позиции разрядов, а затем выполняется логическая операция ИЛИ результата со сдвинутым полем разрядов.

Используемые регистры: AF, BC, D, HL.
Время выполнения: 42 * ПОЗИЦИЯ МЛАДШЕГО РАЗРЯДА плюс 177 тактов (8080) или 40 * ПОЗИЦИЯ МЛАДШЕГО РАЗРЯДА плюс 161 тактов (8085). (По-

зиция младшего разряда определяет, сколько раз маска и поле должны быть сдвинуты влево.)

Размер программы: 52 байта.

Память, необходимая для данных: отсутствует.

Специальные случаи:

1. При попытке записать поле, которое могло бы выйти за конец байта, записываются только разряды вплоть до 7-го. Таким образом, не происходит циклического перехода на начало байта. Если, например, пользователь пытается записать 6-разрядное поле, начиная с разряда 4, то в действительности будут замещены только 4 разряда (разряды с 4-го по 7-й).

2. Как позиция младшего разряда, так и число разрядов в поле интерпретируются по модулю 8. Таким образом, например, позиция разряда 11 эквивалентна позиции разряда 3, а 12-разрядное поле эквивалентно 4-разрядному. Заметим, однако, что число разрядов в поле рассматривается в диапазоне от 1 до 8, т. е., например, 16-разрядное поле эквивалентно 8-разрядному.

3. Попытка записать поле шириной 0 разрядов вызывает возврат с результатом, равным 0.

УСЛОВИЯ НА ВХОДЕ

Данные в регистре A.
Число разрядов в поле (от 1 до 8) в регистре B.
Позиция начального (самого младшего) разряда поля в регистре C.
Поле, которое должно быть вставлено, в регистре D.

УСЛОВИЯ НА ВЫХОДЕ

Результат в регистре A.
Результат равен исходным данным с полем разрядов, вставленным, начиная с заданной позиции.

ПРИМЕРЫ

1. Данные: значение = $F6_{16} = 11110110_2$,
позиция самого младшего разряда = 4,
число разрядов в поле = 2,
поле разрядов = $01_{16} = 00000001_2$.
Результат: значение с вставленным полем = $D6_{16} = 11010110_2$,
в исходное значение было вставлено 2-разрядное поле, начинающееся с разряда 4 (в разряды 4 и 5).

2. Данные: значение = $B8_{16} = 10111000_2$,
позиция самого младшего разряда = 1,
число разрядов в поле = 5,
поле разрядов = $15_{16} = 00010101_2$.
Результат: значение с вставленным полем = $AA_{16} = 10101010_2$,
в исходное значение было вставлено 5-разрядное поле, начинающееся с разряда 1 (в разряды с 1-го по 5-й) изменившее 11100_2 ($1C_{16}$) на 10101_2 (15_{16}).

```
;                                                                    ;
;                                                                    ;
;                                                                    ;
;                                                                    ;
;        ЗАГОЛОВОК:          ЗАПИСЬ ПОЛЯ РАЗРЯДОВ                    ;
;        ИМЯ:                BFI                                     ;
```

```
;                                                                       ;
;                                                                       ;

;                                                                       ;
;       НАЗНАЧЕНИЕ:     ЗАПИСЫВАЕТ ПОЛЕ РАЗРЯДОВ В БАЙТ И ВОЗВРАЩАЕТ    ;
;                       ЭТОТ БАЙТ                                       ;
;                       ЗАМЕЧАНИЕ: ЕСЛИ ТРЕБУЕМОЕ ПОЛЕ СЛИШКОМ ВЕЛИКО,  ;
;                                  ТО БУДУТ ЗАПИСАНЫ ТОЛЬКО РАЗРЯДЫ     ;
;                                  ДО 7-ГО ВКЛЮЧИТЕЛЬНО. НАПРИМЕР, ЕСЛИ ;
;                                  ТРЕБУЕТСЯ ЗАПИСАТЬ 4-РАЗРЯДНОЕ       ;
;                                  ПОЛЕ, НАЧИНАЯ С РАЗРЯДА 7, ТО БУДЕТ  ;
;                                  ЗАПИСАН ТОЛЬКО ОДИН РАЗРЯД (РАЗРЯД   ;
;                                  7).                                  ;
;                                                                       ;
;       ВХОД:           РЕГИСТР A = БАЙТ ДАННЫХ                         ;
;                       РЕГИСТР B = ЧИСЛО РАЗРЯДОВ В ПОЛЕ (1...8)       ;
;                       РЕГИСТР C = НАЧАЛЬНАЯ (МЛАДШАЯ) ПОЗИЦИЯ, В      ;
;                                   КОТОРУЮ ДОЛЖНЫ БЫТЬ ЗАПИСАНЫ ДАННЫЕ ;
;                                   (0...7)                             ;
;                       РЕГИСТР E = ЗАПИСЫВАЕМОЕ ПОЛЕ                   ;
;                                                                       ;
;       ВЫХОД:          РЕГИСТР A = ДАННЫЕ С ЗАПИСАННЫМ ПОЛЕМ           ;
;                                                                       ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF,BC,D,HL                               ;
;                                                                       ;
;       ВРЕМЯ:          177 ТАКТОВ ПЛЮС (42 * ПОЗИЦИЯ МЛАДШЕГО РАЗРЯДА) ;
;                        ТАКТОВ ДЛЯ 8080                                ;
;       ВРЕМЯ:          161 ТАКТ ПЛЮС (40 * ПОЗИЦИЯ МЛАДШЕГО РАЗРЯДА)   ;
;                        ТАКТОВ ДЛЯ 8085                                ;
;                                                                       ;
;       РАЗМЕР:         ПРОГРАММА - 52 БАЙТА                            ;
;                                                                       ;
;                                                                       ;

BFI:
        PUSH    PSW             ;СОХРАНИТЬ ДАННЫЕ

        ;ВЗЯТЬ МАСКУ С ТРЕБУЕМЫМ ЧИСЛОМ НУЛЕВЫХ РАЗРЯДОВ
        PUSH    B               ;СОХРАНИТЬ ПОЗИЦИЮ НАЧАЛЬНОГО РАЗРЯДА
        LXI     H,MSKARY
        MOV     A,B             ;ВЗЯТЬ ЧИСЛО РАЗРЯДОВ (ШИРИНУ ПОЛЯ)
        ANA     A               ;ПРОВЕРИТЬ ШИРИНУ ПОЛЯ
        RZ                      ;ЕСЛИ ШИРИНА ПОЛЯ РАВНА 0, ВЕРНУТЬСЯ
                                ; С ПОЛЕМ, РАВНЫМ 0
        DCR     A               ;ИНДЕКС = ШИРИНА ПОЛЯ - 1
        ANI     00000111B       ;ОГРАНИЧИТЬ ИНДЕКС ДО 0-7
        MOV     C,A
        MVI     B,0
        DAD     B               ;ПОЛУЧИТЬ ИНДЕКС В МАССИВЕ МАСОК
        MOV     D,M             ;D = МАСКА С НУЛЯМИ ДЛЯ ОЧИСТКИ
        POP     B               ;ВОССТАНОВИТЬ НАЧАЛЬНУЮ ПОЗИЦИЮ
        MOV     L,C             ;L = ПОЗИЦИЯ НАЧАЛЬНОГО РАЗРЯДА

        ;ПРОВЕРИТЬ НАЧАЛЬНУЮ ПОЗИЦИЮ НА РАВЕНСТВО НУЛЮ
        MOV     A,C             ;ЧИСЛО СДВИГОВ = ПОЗИЦИЯ
                                ; НАЧАЛЬНОГО РАЗРЯДА В ПОЛЕ
```

```
        ANI     00000111B       ;ОГРАНИЧИТЬ НАЧАЛЬНУЮ ПОЗИЦИЮ ДО 0...7
        MOV     A,E             ;A = ПОЛЕ, КОТОРОЕ ДОЛЖНО БЫТЬ ЗАПИСАНО
        JZ      INSRT           ;ПЕРЕЙТИ, ЕСЛИ НАЧАЛЬНАЯ ПОЗИЦИЯ РАВНА
                                ; НУЛЮ - СДВИГ НЕ НУЖЕН
        ;СДВИНУТЬ ПОЛЕ ДЛЯ ЗАПИСИ
SFIELD:
        ORA     A               ;ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА
        RAL                     ;СДВИНУТЬ ПОЛЕ ЛОГИЧЕСКИ ВЛЕВО.
        DCR     C
        JNZ     SFIELD          ;ПРОДОЛЖАТЬ ДО ОКОНЧАТЕЛЬНОГО СДВИГА
        MOV     E,A             ;E = СДВИНУТОЕ ПОЛЕ

        ;СДВИНУТЬ МАСКИ
        MOV     A,D             ;ВЗЯТЬ МАСКУ
        MOV     C,L             ;ЧИСЛО СДВИГОВ = ПОЗИЦИЯ
                                ; НАЧАЛЬНОГО РАЗРЯДА В ПОЛЕ
SMASK:
        RLC                     ;СДВИНУТЬ ЦИКЛИЧЕСКИ МАСКУ ВЛЕВО
        DCR     C
        JNZ     SMASK           ;ПРОДОЛЖАТЬ ДО ОКОНЧАТЕЛЬНОГО СДВИГА
        MOV     D,A             ;D = СДВИНУТАЯ МАСКА

        ;ВСТАВИТЬ ПОЛЕ
INSRT:
        POP     PSW             ;ВЗЯТЬ ДАННЫЕ ИЗ СТЕКА
        ANA     D               ;ВЫДЕЛИТЬ С ПОМОЩЬЮ МАСКИ
        ORA     E               ;ЗАПИСАТЬ В ПОЛЕ
        RET

        ;МАССИВ МАСОК - ОТ 1 ДО 8 НУЛЕВЫХ РАЗРЯДОВ
MSKARY:
        DB      11111110B
        DB      11111100B
        DB      11111000B
        DB      11110000B
        DB      11100000B
        DB      11000000B
        DB      10000000B
        DB      00000000B


;                                                                    ;
;                                                                    ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                            ;
;                                                                    ;
;                                                                    ;

SC7E:
        MVI     A,0             ;РЕГИСТР A = ДАННЫЕ
        MVI     B,3             ;РЕГИСТР B = ЧИСЛО РАЗРЯДОВ
        MVI     C,2             ;РЕГИСТР C = ПОЗИЦИЯ МЛАДШЕГО РАЗРЯДА
        MVI     E,00000101B     ;РЕГИСТР E = ЗАПИСЫВАЕМОЕ ПОЛЕ
        CALL    BFI             ;ЗАПИСАТЬ 3-РАЗРЯДНОЕ ПОЛЕ, НАЧИНАЯ С
                                ; РАЗРЯДА 2, РЕЗУЛЬТАТ = 00010100B

        JMP     SC7E

        END
```

7F. АРИФМЕТИЧЕСКИЙ СДВИГ ВПРАВО ЧИСЕЛ С ПОВЫШЕННОЙ ТОЧНОСТЬЮ (MPASR)

Многобайтный операнд сдвигается арифметически вправо на заданное число разрядов. Длина операнда 255 байт или меньше. Последний разряд, сдвинутый из самой правой позиции, попадает во флаг переноса. Операнд хранится в памяти таким образом, что его самые младшие по значению байты занимают наименьшие адреса.

Процедура. Знаковый разряд получается из самого старшего по значению байта, сохраняется во флаге переноса, а затем весь операнд сдвигается вправо на один разряд. Эта операция повторяется для заданного числа сдвигов.

Используемые регистры: все.
Время выполнения: ЧИСЛО СДВИГОВ * (52 + 38 * ДЛИНА ОПЕРАНДА В БАЙТАХ) + 65 тактов (8080) или ЧИСЛО СДВИГОВ * (48 + 38 * ДЛИНА ОПЕРАНДА В БАЙТАХ) + 62 такта (8085).
Размер программы: 29 байт.
Память, необходимая для данных: отсутствует.
Специальные случаи:

1. Если длина операнда равна 0, то осуществляется немедленный выход из программы, при этом операнд остается без изменения, а флаг переноса очищается.
2. Если число сдвигов равно 0, то осуществляется немедленный выход из программы, при этом операнд остается без изменения, а флаг переноса очищается.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес операнда в регистрах H и L.
Длина операнда в байтах в регистре B.
Число сдвигов (позиций разрядов) в регистре C.

УСЛОВИЯ НА ВЫХОДЕ

Операнд сдвигается арифметически вправо на заданное число разрядов. Исходный знаковый разряд расширяется вправо. Последний разряд, сдвинутый из самой правой позиции, попадает во флаг переноса. Если число сдвигов или длина операнда равны 0, то флаг переноса очищается.

ПРИМЕРЫ

1. Данные: длина операнда = 08 байт,
операнд = $85A4C719FE06741E_{16}$,
число сдвигов = 04.
Результат: сдвинутый операнд = $F85A4C719FE06741_{16}$,
это исходный операнд, сдвинутый арифметически вправо на четыре разряда; четыре самых старших разряда имеют значения исходного знакового разряда (1),
флаг переноса = 1, так как последний разряд, сдвинутый из самой правой позиции, был равен 1.
2. Данные: длина операнда = 04 байта,
операнд = $3F6A42D3_{16}$,
число сдвигов = 3.

Результат: сдвинутый операнд = $07ED485A_{16}$,
это исходный операнд, сдвинутый арифметически вправо на три разряда; три самых старших разряда имеют значение исходного знакового разряда (0),
флаг переноса = 0, так как последний разряд, сдвинутый из самой правой позиции, был равен 0.

```
;                                                                    ;
;                                                                    ;
;                                                                    ;
;                                                                    ;
;       ЗАГОЛОВОК:      АРИФМЕТИЧЕСКИЙ СДВИГ ВПРАВО ЧИСЕЛ С ПОВЫШЕННОЙ ;
;                       ТОЧНОСТЬЮ                                    ;
;       ИМЯ:            MPASR                                        ;
;                                                                    ;
;                                                                    ;


;                                                                    ;
;       НАЗНАЧЕНИЕ:     СДВИГАЕТ МНОГОБАЙТНЫЕ ОПЕРАНДЫ АРИФМЕТИЧЕСКИ ;
;                       ВПРАВО НА N РАЗРЯДОВ                         ;
;                                                                    ;
;       ВХОД:           РЕГИСТРЫ H И L = БАЗОВЫЙ АДРЕС ОПЕРАНДА      ;
;                       РЕГИСТР B = ДЛИНА ОПЕРАНДА В БАЙТАХ          ;
;                       РЕГИСТР C = ЧИСЛО РАЗРЯДОВ, НА КОТОРОЕ       ;
;                                   ПРОИЗВОДИТСЯ СДВИГ               ;
;                                                                    ;
;                         ARRAY[0] СОДЕРЖИТ МЛАДШИЙ БАЙТ ОПЕРАНДА, А ;
;                         ARRAY[LENGTH-1] - СТАРШИЙ БАЙТ; ЗДЕСЬ ARRAY - ;
;                         БАЗОВЫЙ АДРЕС, А LENGTH - ДЛИНА ОПЕРАНДА   ;
;                                                                    ;
;       ВЫХОД:          ОПЕРАНД СДВИГАЕТСЯ ВПРАВО, ПРИ ЭТОМ ЕГО СТАРШИЙ ;
;                       РАЗРЯД РАЗМНОЖАЕТСЯ                          ;
;                       ФЛАГ ПЕРЕНОСА := ЗНАЧЕНИЕ МЛАДШЕГО РАЗРЯДА,  ;
;                                        СДВИНУТОГО ЗА ПРЕДЕЛЫ ОПЕРАНДА ;
;                                        ПРИ ПОСЛЕДНЕМ СДВИГЕ        ;
;                                                                    ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ                                   ;
;                                                                    ;
;       ВРЕМЯ:          65 ТАКТОВ ПЛЮС                               ;
;                       ((38 * ДЛИНА) + 52) ТАКТОВ НА КАЖДЫЙ СДВИГ ДЛЯ ;
;                       8080                                         ;
;                       62 ТАКТА ПЛЮС                                ;
;                       ((38 * ДЛИНА) + 48) ТАКТОВ НА КАЖДЫЙ СДВИГ ДЛЯ ;
;                       8085                                         ;
;                                                                    ;
;       РАЗМЕР:         ПРОГРАММА - 29 БАЙТ                          ;
;                                                                    ;
;                                                                    ;

MPASR:
        ;ВЫЙТИ, ЕСЛИ ЧИСЛО СДВИГОВ РАВНО 0 ИЛИ ДЛИНА ОПЕРАНДА РАВНА 0
        ;В ЛЮБОМ СЛУЧАЕ КОМАНДЫ ORA ОЧИЩАЮТ ФЛАГ ПЕРЕНОСА
        MOV     A,B
        ORA     A
        RZ                      ;ВЕРНУТЬСЯ, ЕСЛИ ДЛИНА ОПЕРАНДА РАВНА 0
        MOV     A,C
        ORA     A
```

```
        RZ                              ;ВЕРНУТЬСЯ, ЕСЛИ ЧИСЛО СДВИГОВ
                                        ; РАВНО 0

        ;ВЫЧИСЛИТЬ АДРЕС СТАРШЕГО (ПОСЛЕДНЕГО) БАЙТА
        MOV     E,B                     ;E = ДЛИНА ОПЕРАНДА
        MVI     D,0                     ;АДРЕС СТАРШЕГО БАЙТА = БАЗА+ДЛИНА-1
        DAD     D
        DCX     H                       ;HL = АДРЕС СТАРШЕГО БАЙТА
                                        ;C = ЧИСЛО СДВИГОВ

        ;ЦИКЛ ПО ЧИСЛУ СДВИГОВ, КОТОРОЕ ДОЛЖНО БЫТЬ ВЫПОЛНЕНО
        ;НАЧАЛЬНОЕ ЗНАЧЕНИЕ ФЛАГА ПЕРЕНОСА = СТАРШИЙ РАЗРЯД ОПЕРАНДА
LOOP:
        MOV     A,M                     ;ВЗЯТЬ СТАРШИЙ БАЙТ
        RAL                             ;ФЛАГ ПЕРЕНОСА = СТАРШИЙ РАЗРЯД
        MOV     B,E
        PUSH    H                       ;СОХРАНИТЬ АДРЕС СТАРШЕГО РАЗРЯДА

        ;ЦИКЛИЧЕСКИ СДВИНУТЬ БАЙТЫ ВПРАВО, НАЧИНАЯ С САМОГО СТАРШЕГО
ASRLP:
        MOV     A,M
        RAR                             ;ЦИКЛИЧЕСКИ СДВИНУТЬ ВПРАВО СЛЕДУЮЩИЙ
                                        ; БАЙТ
        MOV     M,A
        DCX     H                       ;УМЕНЬШИТЬ АДРЕС ДЛЯ МЛАДШЕГО БАЙТА
        DCR     B
        JNZ     ASRLP
        POP     H                       ;ВОССТАНОВИТЬ АДРЕС СТАРШЕГО БАЙТА
        DCR     C                       ;УМЕНЬШИТЬ ЧИСЛО СДВИГОВ
        JNZ     LOOP
        RET
;                                                                       ;
;                                                                       ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                               ;
;                                                                       ;
;                                                                       ;
SC7F:
        LXI     H,AY                    ;БАЗОВЫЙ АДРЕС ОПЕРАНДА
        MVI     B,SZAY                  ;ДЛИНА ОПЕРАНДА В БАЙТАХ
        MVI     C,SHIFTS                ;ЧИСЛО СДВИГОВ
        CALL    MPASR                   ;АРИФМЕТИЧЕСКИ СДВИНУТЬ ВПРАВО
                        ;РЕЗУЛЬТАТ СДВИГА EDCBA987654321H НА 4 РАЗРЯДА
                        ;          РАВЕН FEDCBA98765432H, C=0
                        ; В ПАМЯТИ AY     = 032H
                        ;          AY1+1 = 054H
                        ;          AY1+2 = 076H
                        ;          AY1+3 = 098H
                        ;          AY1+4 = 0BAH
                        ;          AY1+5 = 0DCH
                        ;          AY1+6 = 0FEH
        JMP     SC7F

        ;СЕКЦИЯ ДАННЫХ
SZAY    EQU     7                       ;ДЛИНА ОПЕРАНДА В БАЙТАХ
SHIFTS  EQU     4                       ;ЧИСЛО СДВИГОВ
AY:     DB      21H,43H,65H,87H,0A9H,0CBH,0EDH

        END
```

7G. ЛОГИЧЕСКИЙ СДВИГ ВЛЕВО ЧИСЕЛ С ПОВЫШЕННОЙ ТОЧНОСТЬЮ (MPLSL)

Многобайтный операнд сдвигается логически влево на заданное число разрядов. Длина операнда 255 байт или меньше. Последний разряд, сдвинутый из самой левой позиции, попадает во флаг переноса. Операнд хранится в памяти таким образом, что его самые *младшие* по значению байты занимают наименьшие адреса.

Процедура. Сначала очищается флаг переноса (чтобы заполнить разряд нулем), а затем весь операнд сдвигается влево на 1 разряд. Эта операция повторяется для заданного числа сдвигов.

Используемые регистры: AF, BC, E.
Время выполнения: ЧИСЛО СДВИГОВ * (45 + 38 * ДЛИНА ОПЕРАНДА В БАЙТАХ) + 43 такта (8080) или ЧИСЛО СДВИГОВ * (41 + 38 * ДЛИНА ОПЕРАНДА В БАЙТАХ) + 39 тактов (8085).
Размер программы: 24 байта.
Память, необходимая для данных: отсутствует.
Специальные случаи:

1. Если длина операнда равна 0, то осуществляется немедленный выход из программы, при этом операнд остается без изменения, а флаг переноса очищается.
2. Если число сдвигов равно 0, то осуществляется немедленный выход из программы, при этом операнд остается без изменения, а флаг переноса очищается.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес операнда в регистрах H и L.
Длина операнда в байтах в регистре B.
Число сдвигов (позиций разрядов) в регистре C.

УСЛОВИЯ НА ВЫХОДЕ

Операнд сдвигается логически влево на заданное число разрядов (самые младшие по значению разряды заполняются нулями). Последний разряд, сдвинутый из самой левой позиции, попадает во флаг переноса. Если число сдвигов равно нулю или длина операнда равна нулю, то флаг переноса очищается.

ПРИМЕРЫ

1. Данные: длина операнда = 08 байт,
   операнд = $85A4C719FE06741E_{16}$,
   число сдвигов = 04.

   Результат: сдвинутый операнд = $5A4C719FE06741E0_{16}$;
   это исходный операнд, сдвинутый логически влево на четыре разряда; четыре самых младших разряда очищаются,
   флаг переноса = 0, так как последний разряд, сдвинутый из самой левой позиции, был равен 0.

2. Данные: длина операнда = 04 байта,
   операнд = $3F6A42D3_{16}$,
   число сдвигов = 03.

   Результат: сдвинутый операнд = $FB521698_{16}$,
   это исходный операнд, сдвинутый логически влево на три разряда; три самых младших разряда очищаются.
   флаг переноса = 1, так как последний разряд, сдвинутый из самой левой позиции, был равен 1.

```
;                                                                      ;
;                                                                      ;
;                                                                      ;
;                                                                      ;
;       ЗАГОЛОВОК:       ЛОГИЧЕСКИЙ СДВИГ ВЛЕВО ЧИСЕЛ С ПОВЫШЕННОЙ    ;
;                        ТОЧНОСТЬЮ                                     ;
;       ИМЯ:             MPLSL                                         ;
;                                                                      ;
;                                                                      ;

;                                                                      ;
;       НАЗНАЧЕНИЕ:      СДВИГАЕТ МНОГОБАЙТНЫЕ ОПЕРАНДЫ ЛОГИЧЕСКИ ВЛЕВО ;
;                        НА N РАЗРЯДОВ                                 ;
;                                                                      ;
;       ВХОД:            РЕГИСТРЫ H И L = БАЗОВЫЙ АДРЕС ОПЕРАНДА       ;
;                        РЕГИСТР B = ДЛИНА ОПЕРАНДА В БАЙТАХ           ;
;                        РЕГИСТР C = ЧИСЛО РАЗРЯДОВ, НА КОТОРОЕ        ;
;                                    ПРОИЗВОДИТСЯ СДВИГ                ;
;                                                                      ;
;                          ARRAY[0] СОДЕРЖИТ МЛАДШИЙ БАЙТ ОПЕРАНДА, А  ;
;                          ARRAY[LENGTH-1] - СТАРШИЙ БАЙТ; ЗДЕСЬ ARRAY - ;
;                          БАЗОВЫЙ АДРЕС, А LENGTH - ДЛИНА ОПЕРАНДА    ;
;                                                                      ;
;       ВЫХОД:           ОПЕРАНД СДВИГАЕТСЯ ВЛЕВО, ПРИ ЭТОМ ЕГО МЛАДШИЕ ;
;                        РАЗРЯДЫ ЗАПОЛНЯЮТСЯ НУЛЯМИ                    ;
;                        ФЛАГ ПЕРЕНОСА := ЗНАЧЕНИЕ СТАРШЕГО РАЗРЯДА,   ;
;                                         СДВИНУТОГО ЗА ПРЕДЕЛЫ ОПЕРАНДА ;
;                                         ПРИ ПОСЛЕДНЕМ СДВИГЕ         ;
;                                                                      ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF,BC,E                                 ;
;                                                                      ;
;       ВРЕМЯ:           43 ТАКТА ПЛЮС                                 ;
;                        ((38 * ДЛИНА) + 45) ТАКТОВ НА КАЖДЫЙ СДВИГ ДЛЯ ;
;                        8080                                          ;
;                        39 ТАКТОВ ПЛЮС                                ;
;                        ((38 * ДЛИНА) + 41) ТАКТОВ НА КАЖДЫЙ СДВИГ ДЛЯ ;
;                        8085                                          ;
;                                                                      ;
;       РАЗМЕР:          ПРОГРАММА - 24 БАЙТА                          ;
;                                                                      ;
;                                                                      ;

MPLSL:
        ;ВЫЙТИ, ЕСЛИ ЧИСЛО СДВИГОВ РАВНО 0 ИЛИ ДЛИНА ОПЕРАНДА РАВНА 0
        ;В ЛЮБОМ СЛУЧАЕ КОМАНДЫ ORA ЧИСТЯТ ФЛАГ ПЕРЕНОСА
        MOV     A,B
        ORA     A
        RZ                      ;ВОЗВРАТИТЬСЯ, ЕСЛИ ДЛИНА ОПЕРАНДА РАВНА 0
        MOV     A,C
        ORA     A
        RZ                      ;ВОЗВРАТИТЬСЯ, ЕСЛИ ЧИСЛО СДВИГОВ
                                ; РАВНО 0

        ;ВЫПОЛНИТЬ В ЦИКЛЕ ЗАДАННОЕ ЧИСЛО СДВИГОВ
        MOV     E,B             ;E = ДЛИНА ОПЕРАНДА
                                ;C = ЧИСЛО СДВИГОВ
                                ;HL = АДРЕС ПЕРВОГО БАЙТА ОПЕРАНДА
```

```
        ;ДЛЯ ЛОГИЧЕСКОГО СДВИГА ЦИКЛ НАЧИНАЕТСЯ С ФЛАГА ПЕРЕНОСА = 0
LOOP:
        PUSH    H               ;СОХРАНИТЬ АДРЕС МЛАДШЕГО БАЙТА
        MOV     B,E             ;B = ДЛИНА ОПЕРАНДА
        ORA     A               ;ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА ДЛЯ ЛОГИЧЕСКОГО
                                ; СДВИГА

        ;СДВИНУТЬ ЦИКЛИЧЕСКИ БАЙТЫ, НАЧИНАЯ С САМОГО МЛАДШЕГО
LSLLP:
        MOV     A,M
        RAL                     ;СДВИНУТЬ ЦИКЛИЧЕСКИ ВЛЕВО СЛЕДУЮЩИЙ БАЙТ
        MOV     M,A
        INX     H               ;ПЕРЕСЛАТЬ В СТАРШИЙ БАЙТ
        DCR     B
        JNZ     LSLLP
        POP     H               ;ВОССТАНОВИТЬ АДРЕС МЛАДШЕГО БАЙТА
        DCR     C               ;УМЕНЬШИТЬ НА 1 ЧИСЛО СДВИГОВ
        JNZ     LOOP
        RET
;                                                                       ;
;                                                                       ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                               ;
;                                                                       ;
;                                                                       ;

SC7G:
        LXI     H,AY            ;БАЗОВЫЙ АДРЕС ОПЕРАНДА
        MVI     B,SZAY          ;ДЛИНА ОПЕРАНДА В БАЙТАХ
        MVI     C,SHIFTS        ;ЧИСЛО СДВИГОВ
        CALL    MPLSL           ;СДВИНУТЬ ЛОГИЧЕСКИ ВЛЕВО
                                ;РЕЗУЛЬТАТ СДВИГА EDCBA987654321H НА
                                ; 4 РАЗРЯДА РАВЕН DCBA9876543210H, C=0
                                ; В ПАМЯТИ AY     = 010H
                                ;         AY1+1 = 032H
                                ;         AY1+2 = 054H
                                ;         AY1+3 = 076H
                                ;         AY1+4 = 098H
                                ;         AY1+5 = 0BAH
                                ;         AY1+6 = 0DCH

        JMP     SC7G

        ;СЕКЦИЯ ДАННЫХ
SZAY    EQU     7               ;ДЛИНА ОПЕРАНДА В БАЙТАХ
SHIFTS  EQU     4               ;ЧИСЛО СДВИГОВ
AY:     DB      21H,43H,65H,87H,0A9H,0CBH,0EDH

        END
```

7H. ЛОГИЧЕСКИЙ СДВИГ ВПРАВО ЧИСЕЛ С ПОВЫШЕННОЙ ТОЧНОСТЬЮ (MPLSR)

Многобайтный операнд сдвигается логически вправо на заданное число разрядов. Длина операнда 255 байт или меньше. Последний разряд, сдвинутый из самой правой позиции, попадает во флаг переноса. Операнд хранится в па-

мяти таким образом, что его самые младшие по значению байты занимают наименьшие адреса.

Процедура. Сначала очищается флаг переноса (для заполнения разряда нулем), а затем весь операнд сдвигается вправо на 1 разряд, начиная со старшего по значению байта. Эта операция повторяется для заданного числа сдвигов.

Используемые регистры: все.
Время выполнения: ЧИСЛО СДВИГОВ * (45 + 38 * ДЛИНА ОПЕРАНДА В БАЙТАХ) + 65 тактов (8080) или ЧИСЛО СДВИГОВ * (41 + 38 * ДЛИНА ОПЕРАНДА В БАЙТАХ) + 62 такта (8085).
Размер программы: 28 байт.
Память, необходимая для данных: отсутствует.
Специальные случаи:

1. Если длина операнда равна 0, то осуществляется немедленный выход из программы, при этом операнд остается без изменения, а флаг переноса очищается.
2. Если число сдвигов равно 0, то осуществляется немедленный выход из программы, при этом операнд остается без изменения, а флаг переноса очищается.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес операнда в регистрах H и L.
Длина операнда в байтах в регистре B.
Число сдвигов (позиций разрядов) в регистре C.

УСЛОВИЯ НА ВЫХОДЕ

Операнд, сдвинутый логически вправо на заданное число позиций. (Самые старшие по значению разряды заполняются нулями).
Последний разряд, сдвинутый из самой правой позиции, попадает во флаг переноса. Если число сдвигов или длина операнда равны нулю, флаг переноса очищается.

ПРИМЕРЫ

1. Данные: длина операнда = 08 байт,
операнд = $85A4C719FE06741E_{16}$,
число сдвигов = 04.
Результат: сдвинутый операнд = $085A4C719FE06741_{16}$;
это исходный операнд, сдвинутый логически вправо на четыре разряда; четыре самых старших разряда очищаются,
флаг переноса = 1, так как последний разряд, сдвинутый из самой правой позиции, был равен 1.
2. Данные: длина операнда = 04 байта,
операнд = $3F6A42D3_{16}$,
число сдвигов = 03.
Результат: сдвинутый операнд = $07ED485A_{16}$,
это исходный операнд, сдвинутый логически вправо на три разряда; три самых старших разряда очищаются,
флаг переноса = 0, так как последний разряд, сдвинутый из самой правой позиции, был равен 0.

```
;                                                                         ;
;                                                                         ;
;                                                                         ;
;                                                                         ;
;       ЗАГОЛОВОК:        ЛОГИЧЕСКИЙ СДВИГ ВПРАВО ЧИСЕЛ С ПОВЫШЕННОЙ       ;
;                         ТОЧНОСТЬЮ                                       ;
;       ИМЯ:              MPLSR                                           ;
;                                                                         ;
;                                                                         ;

;                                                                         ;
;       НАЗНАЧЕНИЕ:       СДВИГАЕТ МНОГОБАЙТНЫЕ ОПЕРАНДЫ ЛОГИЧЕСКИ ВПРАВО,;
;                         НА N РАЗРЯДОВ                                   ;
;                                                                         ;
;       ВХОД:             РЕГИСТРЫ H И L = БАЗОВЫЙ АДРЕС ОПЕРАНДА         ;
;                         РЕГИСТР B = ДЛИНА ОПЕРАНДА В БАЙТАХ             ;
;                         РЕГИСТР C = ЧИСЛО РАЗРЯДОВ, НА КОТОРОЕ          ;
;                                     ПРОИЗВОДИТСЯ СДВИГ                  ;
;                                                                         ;
;                           ARRAY[0] СОДЕРЖИТ МЛАДШИЙ БАЙТ ОПЕРАНДА, А    ;
;                           ARRAY[LENGTH-1] - СТАРШИЙ БАЙТ; ЗДЕСЬ ARRAY - ;
;                           БАЗОВЫЙ АДРЕС, А LENGTH - ДЛИНА ОПЕРАНДА      ;
;                                                                         ;
;       ВЫХОД:            ОПЕРАНД СДВИГАЕТСЯ ВПРАВО, ПРИ ЭТОМ ЕГО СТАРШИЕ ;
;                         РАЗРЯДЫ ЗАПОЛНЯЮТСЯ НУЛЯМИ                      ;
;                         ФЛАГ ПЕРЕНОСА := ЗНАЧЕНИЕ МЛАДШЕГО РАЗРЯДА,     ;
;                                          СДВИНУТОГО ЗА ПРЕДЕЛЫ ОПЕРАНДА ;
;                                          ПРИ ПОСЛЕДНЕМ СДВИГЕ           ;
;                                                                         ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ                                        ;
;                                                                         ;
;       ВРЕМЯ:            65 ТАКТОВ ПЛЮС                                  ;
;                         ((38 * ДЛИНА) + 45) ТАКТОВ НА КАЖДЫЙ СДВИГ ДЛЯ  ;
;                         8080                                            ;
;                         62 ТАКТА ПЛЮС                                   ;
;                         ((38 * ДЛИНА) + 41) ТАКТОВ НА КАЖДЫЙ СДВИГ ДЛЯ  ;
;                         8085                                            ;
;                                                                         ;
;       РАЗМЕР:           ПРОГРАММА - 28 БАЙТ                             ;
;                                                                         ;
;                                                                         ;

MPLSR:
        ;ВЫЙТИ, ЕСЛИ ЧИСЛО СДВИГОВ РАВНО 0 ИЛИ ДЛИНА ОПЕРАНДА РАВНА 0
        ;В ЛЮБОМ СЛУЧАЕ КОМАНДЫ ORA ОЧИЩАЮТ ФЛАГ ПЕРЕНОСА
        MOV     A,B
        ORA     A
        RZ                              ;ВОЗВРАТИТЬСЯ, ЕСЛИ ДЛИНА ОПЕРАНДА РАВНА 0
        MOV     A,C
        ORA     A
        RZ                              ;ВОЗВРАТИТЬСЯ, ЕСЛИ ЧИСЛО СДВИГОВ
                                        ; РАВНО 0

        ;ВЫЧИСЛИТЬ АДРЕС СТАРШЕГО (ПОСЛЕДНЕГО) БАЙТА
        MOV     E,B                     ;E = ДЛИНА ОПЕРАНДА
        MVI     D,0                     ;АДРЕС СТАРШЕГО БАЙТА = БАЗА+ДЛИНА-1
        DAD     D
        DCX     H                       ;HL = АДРЕС СТАРШЕГО БАЙТА
                                        ;C = ЧИСЛО СДВИГОВ
```

```
        ;ВЫПОЛНИТЬ В ЦИКЛЕ ЗАДАННОЕ ЧИСЛО СДВИГОВ
        ;ДЛЯ ЛОГИЧЕСКОГО СДВИГА ЦИКЛ НАЧИНАЕТСЯ С ФЛАГА ПЕРЕНОСА = 0
LOOP:
        ORA     A                       ;ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА ДЛЯ ЛОГИЧЕСКОГО
                                        ; СДВИГА
        MOV     B,E                     ;B = ДЛИНА ОПЕРАНДА
        PUSH    H                       ;СОХРАНИТЬ АДРЕС СТАРШЕГО БАЙТА

        ;СДВИНУТЬ ЦИКЛИЧЕСКИ БАЙТЫ, НАЧИНАЯ С САМОГО СТАРШЕГО
LSRLP:
        MOV     A,M
        RAR                             ;СДВИНУТЬ ЦИКЛИЧЕСКИ ВПРАВО СЛЕДУЮЩИЙ БАЙТ
        MOV     M,A
        DCX     H                       ;ПЕРЕСЛАТЬ В МЛАДШИЙ БАЙТ
        DCR     B
        JNZ     LSRLP
        POP     H                       ;ВОССТАНОВИТЬ АДРЕС СТАРШЕГО БАЙТА
        DCR     C                       ;УМЕНЬШИТЬ НА 1 ЧИСЛО СДВИГОВ
        JNZ     LOOP
        RET

;                                                                       ;
;                                                                       ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                               ;
;                                                                       ;
;                                                                       ;

SC7H:
        LXI     H,AY                    ;HL = БАЗОВЫЙ АДРЕС ОПЕРАНДА
        MVI     B,SZAY                  ;B = ДЛИНА ОПЕРАНДА В БАЙТАХ
        MVI     C,SHIFTS                ;C = ЧИСЛО СДВИГОВ
        CALL    MPLSR                   ;ЛОГИЧЕСКИ СДВИНУТЬ ВПРАВО
                        ;РЕЗУЛЬТАТ СДВИГА EDCBA987654321H НА 4 РАЗРЯДА
                        ;         РАВЕН 0EDCBA98765432H, C=0
                        ; В ПАМЯТИ AY     = 032H
                        ;          AY1+1 = 054H
                        ;          AY1+2 = 076H
                        ;          AY1+3 = 098H
                        ;          AY1+4 = 0BAH
                        ;          AY1+5 = 0DCH
                        ;          AY1+6 = 00EH

        JMP     SC7H

        ;СЕКЦИЯ ДАННЫХ
SZAY    EQU     7                       ;ДЛИНА ОПЕРАНДА В БАЙТАХ
SHIFTS  EQU     4                       ;ЧИСЛО СДВИГОВ
AY:     DB      21H,43H,65H,87H,0A9H,0CBH,0EDH

        END
```

7I. ЦИКЛИЧЕСКИЙ СДВИГ ВПРАВО ЧИСЕЛ С ПОВЫШЕННОЙ ТОЧНОСТЬЮ (MPRR)

Многобайтный операнд сдвигается вправо на заданное число разрядов, так, как если бы самый старший и самый младший по значению разряды были связаны. Длина операнда 255 байт или меньше. Последний разряд, сдвинутый из самой правой позиции, попадает во флаг переноса. Операнд хранится в па-

мяти таким образом, что самые младшие по значению байты занимают наименьшие адреса.

Процедура. Разряд 0 самого младшего по значению байта операнда сдвигается во флаг переноса, а затем весь операнд сдвигается циклически вправо на один разряд, начиная с самого старшего по значению байта. Эта операция повторяется для заданного числа циклических сдвигов.

Используемые регистры: все.
Время выполнения: ЧИСЛО ЦИКЛИЧЕСКИХ СДВИГОВ * (65 + 38 *ДЛИНА ОПЕРАНДА В БАЙТАХ) + 99 тактов (8080) или ЧИСЛО ЦИКЛИЧЕСКИХ СДВИГОВ * (61 + 38 *ДЛИНА ОПЕРАНДА В БАЙТАХ) + 97 тактов (8085).
Размер программы: 38 байт.
Память, необходимая для данных: 1 байт в любом месте ОЗУ для длины операнда (адрес LEN).
Специальные случаи:

1. Если длина операнда равна 0, то осуществляется немедленный выход из программы, при этом операнд остается без изменения, а флаг переноса очищается.
2. Если число циклических сдвигов равно 0, то осуществляется немедленный выход из программы, при этом операнд остается без изменения, а флаг переноса очищается.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес операнда в регистрах H и L.
Длина операнда в байтах в регистре B.
Число циклических сдвигов (позиций разрядов) в регистре C.

УСЛОВИЯ НА ВЫХОДЕ

Операнд, сдвинутый циклически вправо на заданное количество разрядов (самый старший по значению разряд заполняется из самого младшего по значению разряда). Последний разряд, сдвинутый из самой правой позиции, попадает во флаг переноса. Если число циклических сдвигов равно нулю или длина операнда равна нулю, то флаг переноса очищается.

ПРИМЕРЫ

1. Данные: длина операнда = 08,
операнд = $85A4C719FE06741E_{16}$,
число циклических сдвигов = 04.
Результат: операнд, сдвинутый циклически = $E85A4C719FE06741_{16}$,
это исходный операнд, сдвинутый циклически вправо на четыре разряда; четыре самых старших по значению разряда равны четырем начальным младшим по значению разрядам,
флаг переноса = 1, так как последний разряд, сдвинутый из самой правой позиции, был равен 1.
2. Данные: длина операнда = 04 байта,
операнд = $3A6A42D3_{16}$,
число циклических сдвигов = 03.
Результат: операнд, сдвинутый циклически = $67ED485A_{16}$;
это исходный операнд, сдвинутый циклически вправо на три разряда; три самых старших по значению разряда равны трем начальным младшим по значению разрядам,
флаг переноса = 0, так как последний разряд, сдвинутый из самой правой позиции, был равен 0.

```
;                                                                       ;
;                                                                       ;
;                                                                       ;
;                                                                       ;
;       ЗАГОЛОВОК:       ЦИКЛИЧЕСКИЙ СДВИГ ВПРАВО ЧИСЕЛ С ПОВЫШЕННОЙ    ;
;                        ТОЧНОСТЬЮ                                      ;
;       ИМЯ:             MPRR                                           ;
;                                                                       ;
;                                                                       ;

;                                                                       ;
;       НАЗНАЧЕНИЕ:      СДВИГАЕТ МНОГОБАЙТНЫЕ ОПЕРАНДЫ ЦИКЛИЧЕСКИ ВПРАВО;
;                        НА N РАЗРЯДОВ                                  ;
;                                                                       ;
;       ВХОД:            РЕГИСТРЫ H И L = БАЗОВЫЙ АДРЕС ОПЕРАНДА        ;
;                        РЕГИСТР B = ДЛИНА ОПЕРАНДА В БАЙТАХ            ;
;                        РЕГИСТР C = ЧИСЛО РАЗРЯДОВ, НА КОТОРОЕ         ;
;                                    ПРОИЗВОДИТСЯ ЦИКЛИЧЕСКИЙ СДВИГ     ;
;                                                                       ;
;                          ARRAY[0] СОДЕРЖИТ МЛАДШИЙ БАЙТ ОПЕРАНДА, А   ;
;                          ARRAY[LENGTH-1] - СТАРШИЙ БАЙТ; ЗДЕСЬ ARRAY - ;
;                          БАЗОВЫЙ АДРЕС, А LENGTH - ДЛИНА ОПЕРАНДА     ;
;                                                                       ;
;       ВЫХОД:           ОПЕРАНД СДВИГАЕТСЯ ЦИКЛИЧЕСКИ ВПРАВО           ;
;                        ФЛАГ ПЕРЕНОСА := ЗНАЧЕНИЕ ПОСЛЕДНЕГО РАЗРЯДА,  ;
;                                         СДВИНУТОГО ИЗ МЛАДШЕЙ ПОЗИЦИИ ;
;                                                                       ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ                                      ;
;                                                                       ;
;       ВРЕМЯ:           99 ТАКТОВ ПЛЮС                                 ;
;                        ((38 * ДЛИНА) + 65) ТАКТОВ НА КАЖДЫЙ ЦИКЛИЧЕСКИЙ;
;                        СДВИГ ДЛЯ 8080                                 ;
;                        97 ТАКТОВ ПЛЮС                                 ;
;                        ((38 * ДЛИНА) + 61) ТАКТОВ НА КАЖДЫЙ ЦИКЛИЧЕСКИЙ;
;                        СДВИГ ДЛЯ 8085                                 ;
;                                                                       ;
;       РАЗМЕР:          ПРОГРАММА - 38 БАЙТ                            ;
;                        ДАННЫЕ    - 1 БАЙТ                             ;
;                                                                       ;
;                                                                       ;

MPRR:
        ;ВЫЙТИ, ЕСЛИ ЧИСЛО ЦИКЛИЧЕСКИХ СДВИГОВ РАВНО 0 ИЛИ ДЛИНА
        ; ОПЕРАНДА РАВНО 0
        ;В ЛЮБОМ СЛУЧАЕ КОМАНДЫ ORA ОЧИЩАЮТ ФЛАГ ПЕРЕНОСА
        MOV     A,B
        ORA     A
        RZ                      ;ВОЗВРАТИТЬСЯ, ЕСЛИ ДЛИНА ОПЕРАНДА
                                ; РАВНА 0
        STA     LEN             ;СОХРАНИТЬ ДЛИНУ
        MOV     A,C
        ORA     A
        RZ                      ;ВОЗВРАТИТЬСЯ, ЕСЛИ ЧИСЛО
                                ; ЦИКЛИЧЕСКИХ СДВИГОВ РАВНО 0

        ;ВЫЧИСЛИТЬ АДРЕС СТАРШЕГО (ПОСЛЕДНЕГО) БАЙТА
        PUSH    H               ;СОХРАНИТЬ АДРЕС ПЕРВОГО БАЙТА
        MOV     E,B             ;E = ДЛИНА ОПЕРАНДА
```

```
        MVI     D,0                    ;АДРЕС СТАРШЕГО БАЙТА = БАЗА+ДЛИНА-1
        DAD     D
        DCX     H                      ;HL = АДРЕС СТАРШЕГО БАЙТА
        POP     D                      ;DE = АДРЕС МЛАДШЕГО БАЙТА
                                       ;C = ЧИСЛО ЦИКЛИЧЕСКИХ СДВИГОВ

        ;ВЫПОЛНИТЬ В ЦИКЛЕ ЗАДАННОЕ ЧИСЛО ЦИКЛИЧЕСКИХ СДВИГОВ
        ;ФЛАГ ПЕРЕНОСА = МЛАДШИЙ РАЗРЯД ИСХОДНОГО ОПЕРАНДА
LOOP:
        LDAX    D                      ;ВЗЯТЬ МЛАДШИЙ БАЙТ
        RAR                            ;ФЛАГ ПЕРЕНОСА = РАЗРЯД 0 МЛАДШЕГО
                                       ; БАЙТА
        LDA     LEN
        MOV     B,A                    ;B = ДЛИНА ОПЕРАНДА
        PUSH    H                      ;СОХРАНИТЬ АДРЕС СТАРШЕГО БАЙТА

        ;СДВИНУТЬ ЦИКЛИЧЕСКИ БАЙТЫ ВПРАВО, НАЧИНАЯ С САМОГО СТАРШЕГО
RRLP:
        MOV     A,M
        RAR                            ;СДВИНУТЬ ЦИКЛИЧЕСКИ ВПРАВО СЛЕДУЮЩИЙ
                                       ; БАЙТ
        MOV     M,A
        DCX     H                      ;ПРОДОЛЖИТЬ ДЛЯ БОЛЕЕ МЛАДШЕГО БАЙТА
        DCR     B
        JNZ     RRLP
        POP     H                      ;ВОССТАНОВИТЬ АДРЕС СТАРШЕГО БАЙТА
        DCR     C                      ;УМЕНЬШИТЬ НА 1 ЧИСЛО ЦИКЛИЧЕСКИХ
                                       ; СДВИГОВ
        JNZ     LOOP
        RET

LEN:    DS      1                      ;ДЛИНА ОПЕРАНДА В БАЙТАХ


;
;
;       ПРИМЕР ВЫПОЛНЕНИЯ
;
;

SC7I:
        LXI     H,AY                   ;БАЗОВЫЙ АДРЕС ОПЕРАНДА
        MVI     B,SZAY                 ;ДЛИНА ОПЕРАНДА В БАЙТАХ
        MVI     C,ROTATES              ;ЧИСЛО ЦИКЛИЧЕСКИХ СДВИГОВ
        CALL    MPRR                   ;ЦИКЛИЧЕСКИ СДВИНУТЬ ВПРАВО
                                       ;РЕЗУЛЬТАТ ЦИКЛИЧЕСКОГО СДВИГА
                                       ;  EDCBA987654321H НА 4 РАЗРЯДА РАВЕН
                                       ;  1EDCBA98765432H, C=0
                                       ; В ПАМЯТИ AY     = 032H
                                       ;          AY1+1 = 054H
                                       ;          AY1+2 = 076H
                                       ;          AY1+3 = 098H
                                       ;          AY1+4 = 0BAH
                                       ;          AY1+5 = 0DCH
                                       ;          AY1+6 = 01EH

        JMP     SC7I
```

```
        ;СЕКЦИЯ ДАННЫХ
SZAY    EQU     7                       ;ДЛИНА ОПЕРАНДА В БАЙТАХ
ROTATES EQU     4                       ;ЧИСЛО СДВИГОВ
AY:     DB      21H,43H,65H,87H,0A9H,0CBH,0EDH

        END
```

7J. ЦИКЛИЧЕСКИЙ СДВИГ ВЛЕВО ЧИСЕЛ С ПОВЫШЕННОЙ ТОЧНОСТЬЮ (MPRL)

Многобайтный операнд сдвигается влево на заданное число разрядов так, как если бы самый старший и самый младший по значению разряды были связаны. Длина операнда 255 байт или меньше. Последний разряд, сдвинутый из самой левой позиции, попадает во флаг переноса. Операнд хранится в памяти таким образом, что самые младшие по значению байты занимают наименьший адрес.

Процедура. Разряд 7 самого старшего по значению байта операнда сдвигается во флаг переноса, а затем весь операнд сдвигается циклически влево на один разряд, начиная с самого младшего по значению байта. Эта операция повторяется для заданного числа циклических сдвигов.

Используемые регистры: все.
Время выполнения: ЧИСЛО ЦИКЛИЧЕСКИХ СДВИГОВ * (65 + 38 * ДЛИНА ОПЕРАНДА В БАЙТАХ) + 103 такта (8080) или ЧИСЛО ЦИКЛИЧЕСКИХ СДВИГОВ * * (61 + 38 * ДЛИНА ОПЕРАНДА В БАЙТАХ) + 100 тактов (8085).
Размер программы: 39 байт.
Память, необходимая для данных: 1 байт в любом месте ОЗУ для длины операнда (адрес LEN).
Специальные случаи:
1. Если длина операнда равна 0, то осуществляется немедленный выход из программы, при этом операнд остается без изменения, а флаг переноса очищается.
2. Если число циклических сдвигов равно 0, то осуществляется немедленный выход из программы, при этом операнд остается без изменения, а флаг переноса очищается.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес операнда в регистрах H и L.
Длина операнда в байтах в регистре B.
Число циклических сдвигов (позиций разрядов) в регистре C.

УСЛОВИЯ НА ВЫХОДЕ

Операнд, сдвинутый циклически влево на заданное число разрядов (самый младший по значению разряд заполняется из самого старшего по значению разряда). Последний разряд, сдвинутый из самой левой позиции, попадает во флаг переноса. Если число циклических сдвигов равно нулю или длина операнда равна нулю, то флаг переноса очищается.

ПРИМЕРЫ

1. Данные: длина операнда = 08 байт,
операнд = $85A4C719FE06741E_{16}$,
число циклических сдвигов = 04.

Результат: операнд, сдвинутый циклически = $5A4C719FE06741E8_{16}$;
это исходный операнд, сдвинутый циклически влево на четыре разряда; четыре самых младших по значению разряда равны четырем начальным старшим по значению разрядам,
флаг переноса = 0, так как последний разряд, сдвинутый из самой левой позиции, был равен 0.

2. Данные: длина операнда = 04 байта,
операнд = $3F6A42D3_{16}$,
число циклических сдвигов = 03.

Результат: операнд, сдвинутый циклически = $FB521699_{16}$,
это исходный операнд, сдвинутый циклически влево на три разряда; три самых младших по значению разряда равны трем начальным старшим по значению разрядам,
флаг переноса = 1, так как последний разряд, сдвинутый из самой левой позиции, был равен 1.

```
;                                                                   ;
;                                                                   ;
;       ЗАГОЛОВОК:      ЦИКЛИЧЕСКИЙ СДВИГ ВЛЕВО ЧИСЕЛ С ПОВЫШЕННОЙ  ;
;                       ТОЧНОСТЬЮ                                   ;
;       ИМЯ:            MPRL                                        ;
;                                                                   ;
;                                                                   ;


;                                                                   ;
;       НАЗНАЧЕНИЕ:     СДВИГАЕТ МНОГОБАЙТНЫЕ ОПЕРАНДЫ ЦИКЛИЧЕСКИ ВЛЕВО ;
;                       НА N РАЗРЯДОВ                               ;
;                                                                   ;
;       ВХОД:           РЕГИСТРЫ H И L = БАЗОВЫЙ АДРЕС ОПЕРАНДА     ;
;                       РЕГИСТР B = ДЛИНА ОПЕРАНДА В БАЙТАХ         ;
;                       РЕГИСТР C = ЧИСЛО РАЗРЯДОВ, НА КОТОРОЕ      ;
;                                   ПРОИЗВОДИТСЯ ЦИКЛИЧЕСКИЙ СДВИГ  ;
;                                                                   ;
;                         ARRAY[0] СОДЕРЖИТ МЛАДШИЙ БАЙТ ОПЕРАНДА, А ;
;                         ARRAY[LENGTH-1] - СТАРШИЙ БАЙТ; ЗДЕСЬ ARRAY - ;
;                         БАЗОВЫЙ АДРЕС, А LENGTH - ДЛИНА ОПЕРАНДА  ;
;                                                                   ;
;       ВЫХОД:          ОПЕРАНД СДВИГАЕТСЯ ЦИКЛИЧЕСКИ ВЛЕВО         ;
;                       ФЛАГ ПЕРЕНОСА := ЗНАЧЕНИЕ ПОСЛЕДНЕГО РАЗРЯДА, ;
;                                        СДВИНУТОГО ИЗ СТАРШЕЙ ПОЗИЦИИ ;
;                                                                   ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ                                  ;
;                                                                   ;
;       ВРЕМЯ:          103 ТАКТА ПЛЮС                              ;
;                       ((38 * ДЛИНА) + 65) ТАКТОВ НА КАЖДЫЙ СДВИГ ДЛЯ ;
;                       8080                                        ;
;                       100 ТАКТОВ ПЛЮС                             ;
;                       ((38 * ДЛИНА) + 61) ТАКТОВ НА КАЖДЫЙ СДВИГ ДЛЯ ;
;                       8085                                        ;
;                                                                   ;
;       РАЗМЕР:         ПРОГРАММА - 39 БАЙТ                         ;
;                       ДАННЫЕ    - 1 БАЙТ                          ;
;                                                                   ;
;                                                                   ;

MPRL:
        ;ВЫЙТИ, ЕСЛИ ЧИСЛО ЦИКЛИЧЕСКИХ СДВИГОВ РАВНО 0 ИЛИ ДЛИНА
```

```
        ; ОПЕРАНДА РАВНА 0
        ;В ЛЮБОМ СЛУЧАЕ КОМАНДЫ ORA ОЧИЩАЮТ ФЛАГ ПЕРЕНОСА
        MOV     A,B
        ORA     A
        RZ                          ;ВОЗВРАТИТЬСЯ, ЕСЛИ ДЛИНА ОПЕРАНДА
                                    ; РАВНА 0
        STA     LEN                 ;СОХРАНИТЬ ДЛИНУ
        MOV     A,C
        ORA     A
        RZ                          ;ВОЗВРАТИТЬСЯ, ЕСЛИ ЧИСЛО
                                    ; ЦИКЛИЧЕСКИХ СДВИГОВ РАВНО 0

        ;ВЫЧИСЛИТЬ АДРЕС СТАРШЕГО (ПОСЛЕДНЕГО) БАЙТА
        PUSH    H                   ;СОХРАНИТЬ АДРЕС ПЕРВОГО БАЙТА
        MOV     E,B                 ;E = ДЛИНА ОПЕРАНДА
        MVI     D,0                 ;АДРЕС СТАРШЕГО БАЙТА = БАЗА+ДЛИНА-1
        DAD     D
        XCHG
        DCX     D                   ;DE = АДРЕС ПОСЛЕДНЕГО БАЙТА
        POP     H                   ;HL = АДРЕС ПЕРВОГО БАЙТА
                                    ;C = ЧИСЛО ЦИКЛИЧЕСКИХ СДВИГОВ

        ;ВЫПОЛНИТЬ В ЦИКЛЕ ЗАДАННОЕ ЧИСЛО ЦИКЛИЧЕСКИХ СДВИГОВ
        ;ФЛАГ ПЕРЕНОСА = СТАРШИЙ РАЗРЯД ИСХОДНОГО ОПЕРАНДА
LOOP:
        LDAX    D                   ;ВЗЯТЬ СТАРШИЙ ДАЙТ
        RAL                         ;ФЛАГ ПЕРЕНОСА = РАЗРЯД 7 СТАРШЕГО
                                    ; БАЙТА
        LDA     LEN
        MOV     B,A                 ;B = ДЛИНА ОПЕРАНДА В БАЙТАХ
        PUSH    H                   ;СОХРАНИТЬ АДРЕС МЛАДШЕГО БАЙТА

        ;СДВИНУТЬ ЦИКЛИЧЕСКИ БАЙТЫ ВЛЕВО, НАЧИНАЯ С САМОГО МЛАДШЕГО
RLLP:
        MOV     A,M
        RAL                         ;СДВИНУТЬ ЦИКЛИЧЕСКИ ВЛЕВО СЛЕДУЮЩИЙ
                                    ; БАЙТ
        MOV     M,A
        INX     H                   ;ПРОДОЛЖИТЬ ДЛЯ БОЛЕЕ СТАРШЕГО БАЙТА
        DCR     B
        JNZ     RLLP
        POP     H                   ;ВОССТАНОВИТЬ АДРЕС МЛАДШЕГО БАЙТА
        DCR     C                   ;УМЕНЬШИТЬ НА 1 ЧИСЛО ЦИКЛИЧЕСКИХ
                                    ; СДВИГОВ
        JNZ     LOOP
        RET

LEN:    DS      1                   ;ДЛИНА ОПЕРАНДА В БАЙТАХ


;                                                               ;
;                                                               ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                       ;
;                                                               ;
;                                                               ;
```

```
SC7J:
        LXI     H,AY            ;HL = БАЗОВЫЙ АДРЕС ОПЕРАНДА
        MVI     B,SZAY          ;B = ДЛИНА ОПЕРАНДА В БАЙТАХ
        MVI     C,ROTATES       ;C = ЧИСЛО ЦИКЛИЧЕСКИХ СДВИГОВ
        CALL    MPRL            ;ЦИКЛИЧЕСКИ СДВИНУТЬ ВЛЕВО
                                ;РЕЗУЛЬТАТ ЦИКЛИЧЕСКОГО СДВИГА
                                ;  EDCBA987654321H НА 4 РАЗРЯДА РАВЕН
                                ;  DCBA987654321EH, C=0
                                ; В ПАМЯТИ AY     = 01EH
                                ;         AY1+1 = 032H
                                ;         AY1+2 = 054H
                                ;         AY1+3 = 076H
                                ;         AY1+4 = 098H
                                ;         AY1+5 = 0BAH
                                ;         AY1+6 = 0DCH

        JMP     SC7J

        ;СЕКЦИЯ ДАННЫХ
SZAY    EQU     7               ;ДЛИНА ОПЕРАНДА В БАЙТАХ
ROTATES EQU     4               ;ЧИСЛО ЦИКЛИЧЕСКИХ СДВИГОВ
AY:     DB      21H,43H,65H,87H,0A9H,0CBH,0EDH

        END
```

ГЛАВА 8

РАБОТА СО СТРОКАМИ

8А. СРАВНЕНИЕ СТРОК (STRCMP)

Сравниваются две строки и соответствующим образом устанавливаются флаги переноса и нуля. Флаг нуля устанавливается в 1, если строки одинаковые, и в 0 – в противном случае. Флаг переноса устанавливается в 1, если строка с базовым адресом в регистрах D и E (строка 2) больше строки с базовым адресом в регистрах H и L (строка 1). В противном случае флаг переноса устанавливается в 0. Максимальная длина строк равна 255 байтам, при этом действительным символам предшествует байт, содержащий длину. Если на всем протяжении более короткой строки обе строки совпадают, то более длинная строка считается большей.

Процедура. В первую очередь по длинам строк, предшествующим действительным символам, определяется, какая из строк короче. Затем байт за байтом сравниваются строки, вдоль длины, соответствующей самой короткой строке. Если соответствующие байты отличаются, то осуществляется выход из программы и при этом должным образом устанавливаются флаги. Если строки одинаковы вдоль длины самой короткой строки, то флаги устанавливаются при сравнении длины строк.

Используемые регистры: AF, B, DE, HL.
Время выполнения:

1. Если строки не одинаковы вдоль длины самой короткой строки, то время равно 80 + 50 * ЧИСЛО СРАВНИВАЕМЫХ СИМВОЛОВ тактов (8080) или 76 + + 52 * ЧИСЛО СРАВНИВАЕМЫХ СИМВОЛОВ тактов (8085). Например, если до нахождения несовпадения сравниваются пять символов, то время выполнения:

80 + 50 * 5 = 80 + 250 = 330 тактов (8080),
76 + 52 * 5 = 76 + 260 = 336 тактов (8085).

2. Если строки совпадают вдоль длины самой короткой строки, то время равно 120 + 50 * ДЛИНА САМОЙ КОРОТКОЙ СТРОКИ тактов (8080) или 113 + 52 * * ДЛИНА САМОЙ КОРОТКОЙ СТРОКИ тактов (8085). Например, длина самой короткой строки составляет 8 байт, то время выполнения:

120 + 50 * 8 = 128 + 400 = 520 тактов (8080),
113 + 52 * 8 = 113 + 416 = 529 тактов (8085).

Размер программы: 36 байт.
Память, необходимая для данных: 2 байта в любом месте ОЗУ для длин строк (адреса LENS1 и LENS2).

УСЛОВИЯ НА ВХОДЕ

Базовый адрес строки 2 в регистрах D и E.
Базовый адрес строки 1 в регистрах H и L.

УСЛОВИЯ НА ВЫХОДЕ

Флаги устанавливаются так же, как при вычитании строки 2 из строки 1. Если строки одинаковы по всей длине самой короткой строки, то флаги устанавливаются так же, как при вычитании длины строки 2 из длины строки 1.
Флаг нуля = 1, если строки идентичны, и 0, если нет.
Флаг переноса = 1, если строка 2 больше строки 1 и 0, если они идентичны или строка 1 больше. Если строки одинаковы по всей длине самой короткой строки, то считается, что более длинная строка больше.

ПРИМЕРЫ

1. Данные: строка 1 = 05'PRINT' (05 – длина строки)
строка 2 = 03'END' (03 – длина строки).
Результат: флаг нуля = 0 (строки не идентичны),
флаг переноса = 0 (строка 2 не больше строки 1).
2. Данные: строка 1 = 05'PRINT' (05 – длина строки),
строка 2 = 02'PR' (02 – длина строки).
Результат: флаг нуля = 0 (строки не идентичны),
флаг переноса = 0 (строка 2 не больше строки 1).

Считается, что более длинная строка (строка 1) больше. Для того чтобы определить, является ли строка 1 сокращенным вариантом строки 2, следует воспользоваться подпрограммой 8С (поиск позиции подстроки), которая позволяет понять, была ли строка 2, начиная с первого символа, частью строки 1.

3. Данные: строка 1 = 05'PRINT' (05 – длина строки),
строка 2 = 06'SYSTEM' (06 – длина строки).

Результат: флаг нуля = 0 (строки не идентичны),
флаг переноса = 1 (строка 2 больше строки 1).

Здесь считается, что строки содержат символы ASCII. Заметим, что байт, предшествующий действительным символам, содержит шестнадцатеричное число (длину строки), а не символ. Этот байт здесь был представлен в виде двух шестнадцатеричных цифр перед строкой; сама строка заключается в одиночные кавычки.

В этой подпрограмме пробелы рассматриваются так же, как другие символы. Например, если строки содержат символы ASCII, подпрограмма найдет, что SPRINGMAID больше, чем SPRING MAID, так как символ ASCII M ($4D_{16}$) больше, чем пробел ASCII (20_{16}).

```
;                                                                       ;
;                                                                       ;
;                                                                       ;
;                                                                       ;
;       ЗАГОЛОВОК:          СРАВНЕНИЕ СТРОК                             ;
;       ИМЯ:                STRCMP                                      ;
;                                                                       ;
;                                                                       ;


;                                                                       ;
;       НАЗНАЧЕНИЕ:         СРАВНИВАЕТ 2 СТРОКИ И ВОЗВРАЩАЕТ ФЛАГИ C И Z,  ;
;                           РАВНЫМИ 1 ИЛИ 0                             ;
;                                                                       ;
;       ВХОД:               ПАРА РЕГИСТРОВ H = БАЗОВЫЙ АДРЕС СТРОКИ 1   ;
;                           ПАРА РЕГИСТРОВ D = БАЗОВЫЙ АДРЕС СТРОКИ 2   ;
;                                                                       ;
;                             МАКСИМАЛЬНАЯ ДЛИНА СТРОКИ РАВНА 255 БАЙТ ПЛЮС ;
;                             ПРЕДШЕСТВУЮЩИЙ ЕЙ БАЙТ ДЛИНЫ.             ;
;                                                                       ;
;       ВЫХОД:              ЕСЛИ СТРОКА 1 = СТРОКЕ 2, ТО                ;
;                             Z=1, C=0,                                 ;
;                           ЕСЛИ СТРОКА 1 > СТРОКИ 2, ТО                ;
;                             Z=0, C=0,                                 ;
;                           ЕСЛИ СТРОКА 1 < СТРОКИ 2, ТО                ;
;                             Z=0, C=1                                  ;
;                                                                       ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ:  AF, B, DE, HL                           ;
;                                                                       ;
;       ВРЕМЯ:              80 ТАКТОВ ПЛЮС 50 ТАКТОВ НА БАЙТ ПЛЮС 40 ТАКТОВ,;
;                           ЕСЛИ СТРОКИ РАВНЫ, ДЛЯ 8080                 ;
;                           76 ТАКТОВ ПЛЮС 52 ТАКТА НА БАЙТ ПЛЮС 37 ТАКТОВ, ;
;                           ЕСЛИ СТРОКИ РАВНЫ, ДЛЯ 8085                 ;
;                                                                       ;
;       РАЗМЕР:             ПРОГРАММА - 36 БАЙТ                         ;
;                           ДАННЫЕ    -  2 БАЙТА                        ;
;                                                                       ;
;                                                                       ;

STRCMP:
        ;ОПРЕДЕЛИТЬ, КАКАЯ ИЗ СТРОК КОРОЧЕ
        ;ДЛИНА БОЛЕЕ КОРОТКОЙ СТРОКИ = ЧИСЛО БАЙТОВ ДЛЯ СРАВНЕНИЯ
        ; МАССИВА
        LDAX    D                   ;ВЗЯТЬ ДЛИНУ СТРОКИ 2
```

```
        STA     LENS2
        CMP     M               ;СРАВНИТЬ С ДЛИНОЙ СТРОКИ 1
        JC      BEGCMP          ;ПЕРЕЙТИ, ЕСЛИ СТРОКА 2 КОРОЧЕ
        MOV     A,M             ;ИНАЧЕ СТРОКА 1 КОРОЧЕ

        ;СРАВНИТЬ, ИСПОЛЬЗУЯ ДЛИНУ БОЛЕЕ КОРОТКОЙ СТРОКИ
        ;ВЫЙТИ СРАЗУ ЖЕ, КАК ТОЛЬКО ОБНАРУЖИВАЕТСЯ НЕСОВПАДЕНИЕ
        ; СООТВЕТСТВУЮЩИХ СИМВОЛОВ
BEGCMP:
        ORA A                   ;ПРОВЕРИТЬ ДЛИНУ БОЛЕЕ КОРОТКОЙ СТРОКИ
        JZ      CMPLEN          ;СРАВНИТЬ ДЛИНЫ, ЕСЛИ БОЛЕЕ КОРОТКАЯ
                                ; СТРОКА ПУСТАЯ
        MOV     B,A             ;B = ЧИСЛО СРАВНИВАЕМЫХ БАЙТОВ
        XCHG                    ;DE = СТРОКА 1
                                ;HL = СТРОКА 2
        LDAX    D
        STA     LENS1           ;СОХРАНИТЬ ДЛИНУ СТРОКИ 1
CMPLP:
        INX     D               ;ПЕРЕЙТИ К СЛЕДУЮЩИМ БАЙТАМ СТРОК
        INX     H
        LDAX    D               ;ВЗЯТЬ БАЙТ СТРОКИ 1
        CMP     M               ;СРАВНИТЬ С БАЙТОМ СТРОКИ 2
        RNZ                     ;ВЕРНУТЬСЯ С ФЛАГАМИ, РАВНЫМИ 1,
                                ; ЕСЛИ БАЙТЫ НЕ РАВНЫ
        DCR     B
        JNZ     CMPLP

        ;СТРОКИ В ПРЕДЕЛАХ ДЛИНЫ БОЛЕЕ КОРОТКОЙ СТРОКИ СОВПАДАЮТ,
        ; ПОЭТОМУ ДЛЯ УСТАНОВКИ ФЛАГОВ ИСПОЛЬЗОВАТЬ ДЛИНЫ
CMPLEN: LDA     LENS1           ;СРАВНИТЬ ДЛИНЫ
        LXI     H,LENS2
        CMP     M
        RET                     ;ВЕРНУТЬСЯ С ФЛАГАМИ, РАВНЫМИ 1 ИЛИ 0

        ;ДАННЫЕ
LENS1:  DS      1               ;ДЛИНА СТРОКИ 1
LENS2:  DS      1               ;ДЛИНА СТРОКИ 2

;                                                                    ;
;                                                                    ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                            ;
;                                                                    ;
;                                                                    ;

SC8A:
        LXI     H,S1            ;БАЗОВЫЙ АДРЕС СТРОКИ 1
        LXI     D,S2            ;БАЗОВЫЙ АДРЕС СТРОКИ 2
        CALL    STRCMP          ;СРАВНИТЬ СТРОКИ
                                ;В РЕЗУЛЬТАТЕ СРАВНЕНИЯ "STRING 1" И
                                ; "STRING 2" ПОЛУЧАЕМ, ЧТО СТРОКА 1
                                ; КОРОЧЕ СТРОКИ 2, ПОЭТОМУ Z=0,C=1

        JMP     SC8A            ;ЦИКЛ ДЛЯ СЛЕДУЮЩЕГО ТЕСТА

S1:     DB      20H,'STRING 1                          '
S2:     DB      20H,'STRING 2                          '

        END
```

8В. ОБЪЕДИНЕНИЕ СТРОК (CONCAT)

Объединяются две строки с размещением в памяти второй строки сразу за первой. Если в результате объединения строка получается длиннее строки заданного максимального размера, то присоединяется только та часть строки 2, которая дает возможность получить объединенную строку максимальной длины. В случае, когда может быть присоединена вся строка 2, флаг переноса очищается, если же должна быть отброшена часть строки 2, флаг переноса устанавливается в 1. Максимальная длина каждой строки 255 байт, при этом действительным символам предшествует байт, содержащий длину.

Процедура. Для определения места, с которого следует начинать добавлять символы, в программе используется длина строки 1, а для определения числа символов, которые необходимо добавить – длина строки 2. Если сумма этих длин превышает максимум, программа указывает на переполнение и уменьшает число символов, которые должны быть добавлены (это число равно максимальной длине минус длина строки 1). Затем соответствующее число символов пересылается из строки 2 в конец строки 1, обновляется длина строки 1 и устанавливается флаг переноса, указывающий, были ли отброшены какие-либо символы.

Используемые регистры: все.

Время выполнения: приблизительно 39 * ЧИСЛО ПРИСОЕДИНЯЕМЫХ СИМВОЛОВ плюс 279 тактов (8080) или 40 * ЧИСЛО ПРИСОЕДИНЯЕМЫХ СИМВОЛОВ плюс 265 тактов (8085). ЧИСЛО ПРИСОЕДИНЯЕМЫХ СИМВОЛОВ обычно равно длине строки 2, но если объединенная строка может быть слишком длинной, то равно максимальной длине строки 1 минус ее текущая длина. Например, если ЧИСЛО ПРИСОЕДИНЯЕМЫХ СИМВОЛОВ равно 14_{16} (20_{10}), то время выполнения

$39 * 20 + 279 = 780 + 279 = 1059$ тактов (8080),

$40 * 20 + 265 = 800 + 265 = 1065$ тактов (8085).

Размер программы: 83 байта.

Память, необходимая для данных: 5 байт в любом месте ОЗУ для базового адреса строки 1 (2 байта, начиная с адреса S1ADR), длин строк (адреса S1LEN и S2LEN) и флага, указывающего на переполнение при соединении строк (адреса STRGOV).

Специальные случаи:

1. Если объединение может привести к тому, что длина строки превысит заданную максимальную длину, то присоединяется только та часть строки 2, которая позволяет достичь максимума. Если строка 2 должна быть усечена, флаг переноса устанавливается в 1.

2. Если строка 2 имеет длину 0, то осуществляется выход из программы с очищенным флагом переноса (нет ошибок), при этом строка 1 остается без изменения. Таким образом, длина 0 для любой из строк интерпретируется как 0, а не как 256.

3. Если начальная длина строки 1 превышает заданный максимум, то осуществляется выход из программы с флагом переноса, установленным в 1 (что указывает на ошибку), при этом строка 1 остается без изменения.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес строки 2 в регистрах D и E.

Базовый адрес строки 1 в регистрах H и L.
Максимальная длина строки 1 в регистре B.

УСЛОВИЯ НА ВЫХОДЕ

Строка 2 присоединена к концу строки 1, при этом длина строки 1 соответствующим образом увеличена. Если длина результирующей строки может превысить максимальную, то присоединяется только та часть строки 2, которая позволяет получить строку 1 максимальной длины. Если какая-либо часть строки 2 должна быть отброшена, то флаг переноса устанавливается в 1. В противном случае флаг переноса очищается.

ПРИМЕРЫ

1. Данные: максимальная длина строки 1 = $0E_{16} = 14_{10}$,
строка 1 = 07'JOHNSON' (07 – длина строки),
строка 2 = 05' , DON' (05 – длина строки).
Результат: строка 1 = 0C'JOHNSON', DON' ($0C_{16} = 12_{10}$ – длина строки, полученной в результате объединения строки 2 со строкой 1),
флаг переноса = 0, так как в результате объединения длина строки не превысила максимальную.
2. Данные: максимальная длина строки 1 = $0E_{16} = 14_{10}$,
строки 1 = 07'JOHNSON' (07 – длина строки),
строка 2 = 09',RICHARD' (09 – длина строки).
Результат: строка 1 = 0E'JOHNSON, RICHA' ($0E_{16} = 14_{10}$ – максимальная допустимая длина, так что последние два символа строки 2 были отброшены),
флаг переноса = 1, так как в результате объединения получается строка, длина которой превышает максимальную.

Заметим, что в обоих примерах начальный байт (содержащий длину строки) представлен в виде двух шестнадцатеричных цифр.

```
;                                                                ;
;                                                                ;
;                                                                ;
;                                                                ;
;       ЗАГОЛОВОК:      ОБ'ЕДИНЕНИЕ СТРОК                        ;
;       ИМЯ:            CONCAT                                   ;
;                                                                ;
;                                                                ;


;                                                                ;
;       НАЗНАЧЕНИЕ:     ОБ'ЕДИНЯЕТ ДВЕ СТРОКИ В ОДНУ             ;
;                                                                ;
;       ВХОД:           ПАРА РЕГИСТРОВ H = БАЗОВЫЙ АДРЕС СТРОКИ 1 ;
;                       ПАРА РЕГИСТРОВ D = БАЗОВЫЙ АДРЕС СТРОКИ 2 ;
;                       РЕГИСТР B = МАКСИМАЛЬНАЯ ДЛИНА СТРОКИ 1  ;
;                                                                ;
;                         МАКСИМАЛЬНАЯ ДЛИНА СТРОКИ 255 БАЙТ ПЛЮС ОДИН ;
;                         ПРЕДШЕСТВУЮЩИЙ БАЙТ С ДЛИНОЙ СТРОКИ     ;
;                                                                ;
;       ВЫХОД:          СТРОКА 1 := СТРОКА 1, ОБ'ЕДИНЕННАЯ СО СТРОКОЙ 2 ;
;                       ЕСЛИ НЕТ ОШИБОК, ТО                      ;
;                         ФЛАГ ПЕРЕНОСА := 0                     ;
;                       ИНАЧЕ                                    ;
;                         ФЛАГ ПЕРЕНОСА := 1.                    ;
;                         ЕСЛИ В РЕЗУЛЬТАТЕ ОБ'ЕДИНЕНИЯ ДЛИНА СТРОКИ 1 ;
```

```
;                            ПОЛУЧАЕТСЯ БОЛЬШЕ МАКСИМАЛЬНОЙ, ТО ДОБАВЛЯЕТСЯ;
;                            ТОЛЬКО ЧАСТЬ СТРОКИ 2, И В РЕЗУЛЬТАТЕ СТРОКА  ;
;                            1 ИМЕЕТ СВОЮ МАКСИМАЛЬНУЮ ДЛИНУ.              ;
;                            ЕСЛИ ДЛИНА СТРОКИ 1 > МАКСИМАЛЬНОЙ ДЛИНЫ, ТО  ;
;                            ДОБАВЛЕНИЯ НЕ ПРОИСХОДИТ.                     ;
;                                                                          ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ                                         ;
;                                                                          ;
;       ВРЕМЯ:              ПРИБЛИЗИТЕЛЬНО 39 * (ДЛИНА СТРОКИ 2) ТАКТОВ    ;
;                           ПЛЮС 279 ТАКТОВ ДЛЯ 8080                       ;
;                           ПРИБЛИЗИТЕЛЬНО 40 * (ДЛИНА СТРОКИ 2) ТАКТОВ    ;
;                           ПЛЮС 265 ТАКТОВ ДЛЯ 8085                       ;
;                                                                          ;
;       РАЗМЕР:             ПРОГРАММА - 89 БАЙТ                            ;
;                           ДАННЫЕ    -  5 БАЙТ                            ;
;                                                                          ;
;                                                                          ;

CONCAT:
        ;ОПРЕДЕЛИТЬ, ГДЕ НАЧАТЬ ОБ'ЕДИНЕНИЕ
        ;ОБ'ЕДИНЕНИЕ НАЧИНАЕТСЯ С КОНЦА СТРОКИ 1
        ;КОНЕЦ СТРОКИ 1 = БАЗА1 + ДЛИНА1 + 1, ГДЕ 1 ДОБАВЛЯЕТСЯ ДЛЯ
        ; БАЙТА ДЛИНЫ
        ;НОВЫЕ СИМВОЛЫ ИЗ СТРОКИ 2 БЕРУТСЯ, НАЧИНАЯ С АДРЕСА БАЗА2 + 1
        ; (ПРИ ЭТОМ ПРОПУСКАЕТСЯ БАЙТ ДЛИНЫ)
        SHLD    S1ADR           ;СОХРАНИТЬ АДРЕС СТРОКИ 1
        PUSH    B               ;СОХРАНИТЬ МАКСИМАЛЬНУЮ ДЛИНУ СТРОКИ 1
        MOV     A,M             ;СОХРАНИТЬ ДЛИНУ СТРОКИ 1
        STA     S1LEN
        MOV     C,A             ;КОНЕЦ1 = БАЗА1 + ДЛИНА1
        MVI     B,0
        DAD     B               ;HL = ПОСЛЕДНИЙ СИМВОЛ В СТРОКЕ 1
        LDAX    D               ;СОХРАНИТЬ ДЛИНУ СТРОКИ 2
        STA     S2LEN
        POP     B               ;ВОССТАНОВИТЬ МАКСИМАЛЬНУЮ ДЛИНУ

        ;ОПРЕДЕЛИТЬ, СКОЛЬКО СИМВОЛОВ НЕОБХОДИМО ПРИСОЕДИНИТЬ
        ;ЭТА ВЕЛИЧИНА РАВНА ДЛИНЕ СТРОКИ 2, ЕСЛИ ДЛИНА ОБ'ЕДИНЕННЫХ
        ; СТРОК НЕ ПРЕВЫШАЕТ МАКСИМАЛЬНУЮ
        ;ИНАЧЕ ОНА РАВНА МАКСИМАЛЬНОМУ ЧИСЛУ БАЙТОВ, ПРИ КОТОРОМ
        ; ОБ'ЕДИНЕННАЯ СТРОКА ДОСТИГАЕТ СВОЕЙ МАКСИМАЛЬНОЙ ДЛИНЫ
        ; МИНУС ДЛИНА СТРОКИ 1
        MOV     C,A             ;C = ДЛИНА СТРОКИ 2
        LDA     S1LEN
        ADD     C               ;СЛОЖИТЬ ДЛИНЫ СТРОК
        JC      TOOLNG          ;ПЕРЕЙТИ, ЕСЛИ СУММА ПРЕВЫШАЕТ 255
        CMP     B               ;СРАВНИТЬ С МАКСИМАЛЬНОЙ ДЛИНОЙ
        JZ      LENOK           ;ПЕРЕЙТИ, ЕСЛИ ДЛИНА НОВОЙ СТРОКИ РАВНА
        JC      LENOK           ; МАКСИМАЛЬНОЙ ДЛИНЕ ИЛИ МЕНЬШЕ ЕЕ

        ;ОБ'ЕДИНЕННАЯ СТРОКА СЛИШКОМ ДЛИННАЯ
        ; УСТАНОВИТЬ ФЛАГ ПЕРЕПОЛНЕНИЯ СТРОКИ STRGOV := 0FFH
        ; УСТАНОВИТЬ ЧИСЛО ПРИСОЕДИНЯЕМЫХ СИМВОЛОВ =
        ;  МАКСИМАЛЬНАЯ ДЛИНА - S1LEN
        ; УСТАНОВИТЬ ДЛИНУ СТРОКИ 1 РАВНОЙ МАКСИМАЛЬНОЙ ДЛИНЕ
TOOLNG:
        MVI     A,0FFH          ;УСТАНОВИТЬ ФЛАГ ПЕРЕПОЛНЕНИЯ СТРОКИ
```

```
        STA     STRGOV
        LDA     S1LEN           ;ВЫЧИСЛИТЬ ДЛИНУ СТРОКИ 1 =
        MOV     C,A             ; МАКСИМАЛЬНАЯ ДЛИНА - S1LEN
        MOV     A,B
        SUB     C
        RC                      ;ВЫЙТИ, ЕСЛИ ИСХОДНАЯ СТРОКА СЛИШКОМ
                                ; ВЕЛИКА
        STA     S2LEN           ;ИЗМЕНИТЬ S2LEN НА ЗНАЧЕНИЕ, РАВНОЕ
                                ; МАКСИМАЛЬНАЯ ДЛИНА - S1LEN
        MOV     A,B             ;ДЛИНА СТРОКИ 1 = МАКСИМАЛЬНАЯ ДЛИНА
        STA     S1LEN
        JMP     DOCAT           ;ВЫПОЛНИТЬ ОБ'ЕДИНЕНИЕ

        ;ОБ'ЕДИНЁННАЯ СТРОКА НЕ ПРЕВЫШАЕТ МАКСИМАЛЬНУЮ ДЛИНУ
        ; ДЛИНА СТРОКИ 1 = S1LEN + S2LEN
        ; УСТАНОВИТЬ ФЛАГ ОТСУТСТВИЯ ПЕРЕПОЛНЕНИЯ STRGOV := 0
        ; ЧИСЛО СИМВОЛОВ, КОТОРОЕ НЕОБХОДИМО ДОБАВИТЬ =
        ;  ДЛИНА СТРОКИ 2
LENOK:
        STA     S1LEN           ;СОХРАНИТЬ СУММУ ДЛИН
        SUB     A
        STA     STRGOV          ;УСТАНОВИТЬ ФЛАГ ОТСУТСТВИЯ ПЕРЕПОЛНЕНИЯ

        ;ОБ'ЕДИНЕНИЕ СТРОК
DOCAT:
        LDA     S2LEN           ;ВЗЯТЬ ЧИСЛО СИМВОЛОВ
        ORA     A
        JZ      EXIT            ;ВЫЙТИ, ЕСЛИ НЕЧЕГО ДОБАВЛЯТЬ
        MOV     B,A             ;B = ЧИСЛО СИМВОЛОВ
                                ;HL = АДРЕС НАЗНАЧЕНИЯ
                                ;DE = ИСХОДНЫЙ АДРЕС
CATLP:
        INX     H               ;УВЕЛИЧИТЬ АДРЕСА ДЛЯ СЛЕДУЮЩИХ
        INX     D               ; СИМВОЛОВ
        LDAX    D               ;ВЗЯТЬ СИМВОЛ ИЗ СТРОКИ 2
        MOV     M,A             ;ПЕРЕСЛАТЬ ЕГО В КОНЕЦ СТРОКИ 1
        DCR     B
        JNZ     CATLP           ;ПРОДОЛЖАТЬ ДЛЯ ВСЕХ СИМВОЛОВ

EXIT:
        LDA     S1LEN           ;ВЗЯТЬ НОВУЮ ДЛИНУ СТРОКИ 1
        LHLD    S1ADR           ;ВЗЯТЬ АДРЕС ДЛИНЫ СТРОКИ 1
        MOV     M,A             ;УСТАНОВИТЬ ДЛИНУ
        LDA     STRGOV
        RAR                     ;ФЛАГ ПЕРЕНОСА = 1, ЕСЛИ ЕСТЬ
                                ; ПЕРПОЛНЕНИЕ, 0 - ЕСЛИ НЕТ
        RET

        ;ДАННЫЕ
S1ADR:  DS      2               ;БАЗОВЫЙ АДРЕС СТРОКИ 1
S1LEN:  DS      1               ;ДЛИНА СТРОКИ 1
S2LEN:  DS      1               ;ДЛИНА СТРОКИ 2
STRGOV: DS      1               ;ФЛАГ ПЕРЕПОЛНЕНИЯ СТРОКИ


;                                                                  ;
;                                                                  ;
```

```
;       ПРИМЕР ВЫПОЛНЕНИЯ                                          ;
;                                                                  ;
;                                                                  ;

SC8B:
        LXI     H,S1            ;HL = БАЗОВЫЙ АДРЕС S1
        LXI     D,S2            ;DE = БАЗОВЫЙ АДРЕС S2
        MVI     B,20H           ;B = МАКСИМАЛЬНАЯ ДЛИНА СТРОКИ 1
        CALL    CONCAT          ;ОБ'ЕДИНИТЬ СТРОКИ

        JMP     SC8B            ;РЕЗУЛЬТАТ ОБ'ЕДИНЕНИЯ
                                ; "LASTNAME" И ", FIRSTNAME" РАВЕН
                                ; S1 = 13H,"LASTNAME, FIRSTNAME"

        ;ДАННЫЕ ДЛЯ ТЕСТА, КОТОРЫЕ МОЖНО ИЗМЕНИТЬ НА ДРУГИЕ ЗНАЧЕНИЯ
S1:     DB      8H              ;ДЛИНА S1
        DB      'LASTNAME                           ' ;МАКС. ДЛИНА 32 БАЙТА
S2:     DB      0BH             ;ДЛИНА S2
        DB      ', FIRSTNAME                        ' ;МАКС. ДЛИНА 32 БАЙТА

        END
```

8С. ПОИСК ПОЗИЦИИ ПОДСТРОКИ (POS)

Ищется первое появление подстроки внутри строки. Если подстрока найдена, возвращается индекс, с которого она начинается, а если не найдена – 0. Максимальная длина как строки, так и подстроки, 255 байт, при этом действительным символам предшествует байт, содержащий длину. Таким образом, если подстрока найдена, ее начальный индекс не может быть меньше 1 или больше 255.

Процедура. Просматривается строка до тех пор, пока не находится подстрока или же пока оставшаяся часть строки не будет короче подстроки и, следовательно, не сможет содержать подстроку. Если строка не содержит подстроку, то очищается аккумулятор; в противном случае в аккумулятор помещается начальный индекс подстроки.

Используемые регистры: все.

Время выполнения: зависит от данных, однако дополнительное время составляет 165 тактов (8080) или 154 такта (8085), каждое успешное сравнение одного символа занимает 59 тактов (8080) или 54 такта (8085), а каждое безуспешное сравнение одного символа занимает 133 такта (8080 или 8085). Наихудшим будет тот случай, когда строка и подстрока совпадают по всей длине, за исключением последнего символа подстроки, такой как

строка = 'AAAAAAAAB'
подстрока = 'AAB',

Время выполнения в этом случае будет

(ДЛИНА СТРОКИ – ДЛИНА ПОДСТРОКИ + 1) * (59 * (ДЛИНА ПОДСТРОКИ – – 1) + 133) + 165 (8080)
(ДЛИНА СТРОКИ – ДЛИНА ПОДСТРОКИ + 1) * (54 * (ДЛИНА ПОДСТРОКИ – – 1) + 133) + 154 (8085)

Например, если ДЛИНА СТРОКИ = 9 и ДЛИНА ПОДСТРОКИ = 3 (как в показанном случае), то время выполнения будет:

(9 – 3 + 1) * (59 * (3 – 1) + 133) + 165 = 7 * 251 + 165 = 1757 + 165 = 1922 такта (8080),
(9 – 3 + 1) * (54 * (3 – 1) + 133) + 154 = 7 * 241 + 154 = 1687 + 154 = 1841 такт (8085).

Размер программы: 81 байт.

Память, необходимая для данных: 7 байт в любом месте ОЗУ для базового адреса строки (2 байта, начиная с адреса STRING), базового адреса подстроки (2 байта, начиная с адреса SUBSTG), длины строки (адрес SLEN), длины подстроки (адрес SUBLEN) и текущего начального индекса в строке (адрес INDEX).

Специальные случаи:

1. Если длина строки или подстроки составляет 0, то осуществляется выход из программы с нулем в аккумуляторе, указывающем, что подстрока не найдена.
2. Если подстрока длиннее строки, то осуществляется выход из программы с нулем в аккумуляторе, указывающем, что подстрока не найдена.
3. Если возвращается индекс, равный 1, то подстрока может рассматриваться как сокращенный вариант строки, т. е. подстрока содержится в строке, начиная с первого символа. Типичным может быть пример строки PRINT и подстроки PR.
4. Если подстрока содержится в строке более одного раза, то возвращается только индекс первого появления (появления с наименьшим начальным индексом).

УСЛОВИЯ НА ВХОДЕ

Базовый адрес подстроки в регистрах D и E.
Базовый адрес в регистрах H и L.

УСЛОВИЯ НА ВЫХОДЕ

Если подстрока найдена, то регистр A содержит индекс первого появления подстроки в строке, если же подстрока не найдена, то регистр A содержит 0.

ПРИМЕРЫ

1. Данные: строка 1D´ENTER SPEED IN MILES PER HOUR´ ($1D_{16} = 29_{10}$ – длина строки), подстрока = 05´MILES´ (05 – длина подстроки).
   Результат: регистр A содержит 10_{16} (16_{10}) – индекс, с которого начинается подстрока ´MILES´.
2. Данные: строка = 1B´SALES FIGURES FOR JUNE 1981´ ($1B_{16} = 27_{10}$ – длина строки), подстрока = 04´JUNE´ (04 – длина подстроки).
   Результат: регистр A содержит 13_{16} (19_{10}) – индекс, с которого начинается подстрока ´JUNE´
3. Данные: строка = 10´LET Y1 = X1 + R7´ ($10_{16} = 16_{10}$ – длина строки), подстрока = 02´R4´(02 – длина подстроки).
   Результат: регистр A содержит 0, так как подстрока ´R4´ не появляется в строке LET Y1 = X1 + R7.
4. Данные: строка = 07´RESTORE´ (07 – длина строки), подстрока = 03´RES´ (03 – длина подстроки).
   Результат: регистр A содержит 1 – индекс, с которого начинается подстрока ´RES´. Индекс 1 указывает, что подстрока, возможно, является сокращенным вариантом строки. В интерактивных программах, таких как интерпретаторы BASIC и процессоры слов, часто используются подобные сокращения для упрощения ввода и экономии памяти.

```
;                                                                       ;
;                                                                       ;
;                                                                       ;
;                                                                       ;
;       ЗАГОЛОВОК:      ПОИСК ПОЗИЦИИ ПОДСТРОКИ                         ;
;       ИМЯ:            POS                                             ;
;                                                                       ;
;                                                                       ;

;                                                                       ;
;       НАЗНАЧЕНИЕ:     ИЩЕТ ПЕРВОЕ ПОЯВЛЕНИЕ ПОДСТРОКИ В СТРОКЕ И      ;
;                       ВОЗВРАЩАЕТ ЕЕ НАЧАЛЬНЫЙ ИНДЕКС. ЕСЛИ ПОДСТРОКА  ;
;                       НЕ НАЙДЕНА, ТО ВОЗВРАЩАЕТ 0                     ;
;                                                                       ;
;       ВХОД:           ПАРА РЕГИСТРОВ H = БАЗОВЫЙ АДРЕС СТРОКИ         ;
;                       ПАРА РЕГИСТРОВ D = БАЗОВЫЙ АДРЕС ПОДСТРОКИ      ;
;                                                                       ;
;                         МАКСИМАЛЬНАЯ ДЛИНА СТРОКИ 255 БАЙТ ПЛЮС ОДИН  ;
;                         ПРЕДШЕСТВУЮЩИЙ БАЙТ С ДЛИНОЙ СТРОКИ.          ;
;                                                                       ;
;       ВЫХОД:          ЕСЛИ ПОДСТРОКА НАЙДЕНА, ТО                      ;
;                         РЕГИСТР A = ЕЕ НАЧАЛЬНЫЙ ИНДЕКС               ;
;                       ИНАЧЕ                                           ;
;                         РЕГИСТР A = 0                                 ;
;                                                                       ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ                                      ;
;                                                                       ;
;       ВРЕМЯ:          ТАК КАК АЛГОРИТМ ЗАВИСИТ ОТ ДАННЫХ, ТО          ;
;                       НЕВОЗМОЖНО ДАТЬ ПРОСТУЮ ФОРМУЛУ, ОДНАКО         ;
;                       СЛЕДУЮЩИЕ СООБРАЖЕНИЯ ЯВЛЯЮТСЯ ПРАВИЛЬНЫМИ:     ;
;                                                                       ;
;                       165 ТАКТОВ ПЛЮС 59 ТАКТОВ НА КАЖДОЕ СОВПАДЕНИЕ  ;
;                       ОДНОГО СИМВОЛА, 133 ТАКТА - НА НЕСОВПАДЕНИЕ,    ;
;                       ДЛЯ 8080.                                       ;
;                       154 ТАКТА ПЛЮС 54 ТАКТА НА КАЖДОЕ СОВПАДЕНИЕ    ;
;                       ОДНОГО СИМВОЛА, 133 ТАКТА - НА НЕСОВПАДЕНИЕ,    ;
;                       ДЛЯ 8085.                                       ;
;                                                                       ;
;                       НАИХУДШИМ ПО ВРЕМЕНИ ВЫПОЛНЕНИЯ БУДЕТ ТАКОЙ     ;
;                       СЛУЧАЙ, КОГДА СТРОКА И ПОДСТРОКА ЦЕЛИКОМ        ;
;                       СОВПАДАЮТ, КРОМЕ ПОСЛЕДНЕГО СИМВОЛА ПОДСТРОКИ,  ;
;                       НАПРИМЕР:                                       ;
;                           СТРОКА = 'AAAAAAAAB'                        ;
;                           ПОДСТРОКА = 'AAB'                           ;
;                                                                       ;
;       РАЗМЕР:         ПРОГРАММА - 81 БАЙТ                             ;
;                       ДАННЫЕ    -  7 БАЙТ                             ;
;                                                                       ;
;                                                                       ;

POS:
        ;УСТАНОВИТЬ РАБОЧИЕ ЯЧЕЙКИ
        ;ВЫЙТИ, ЕСЛИ СТРОКА ИЛИ ПОДСТРОКА ИМЕЕТ НУЛЕВУЮ ДЛИНУ
        SHLD    STRING          ;СОХРАНИТЬ АДРЕС СТРОКИ
        XCHG
        MOV     A,M             ;ПРОВЕРИТЬ ДЛИНУ ПОДСТРОКИ
        ORA     A
        JZ      NOTFND          ;ВЫЙТИ, ЕСЛИ ДЛИНА ПОДСТРОКИ = 0
```

```
        INX     H                       ;ПЕРЕЙТИ НА БАЙТ ПОСЛЕ ДЛИНЫ
                                        ; ПОДСТРОКИ
        SHLD    SUBSTG                  ;СОХРАНИТЬ АДРЕС ПОДСТРОКИ
        STA     SUBLEN
        MOV     C,A                     ;C = ДЛИНА ПОДСТРОКИ
        LDAX    D
        ORA A                           ;ПРОВЕРИТЬ ДЛИНУ СТРОКИ
        JZ      NOTFND                  ;ВЫЙТИ, ЕСЛИ ДЛИНА СТРОКИ = 0

        ;ЧИСЛО ПОИСКОВ = ДЛИНА СТРОКИ - ДЛИНА ПОДСТРОКИ + 1.
        ;ПОСЛЕ ЭТОГО ПОИСК НЕ ИСПОЛЬЗУЕТСЯ, ТАК КАК ОСТАЕТСЯ
        ;НЕДОСТАТОЧНО БАЙТОВ ПО СРАВНЕНИЮ С ДЛИНОЙ ПОДСТРОКИ
        ;
        ;ЕСЛИ ПОДСТРОКА ДЛИННЕЕ СТРОКИ, ТО НЕМЕДЛЕННО ВЫЙТИ,
        ; УКАЗАВ, ЧТО ПОДСТРОКА НЕ НАЙДЕНА
        SUB     C                       ;A = ДЛИНА СТРОКИ - ДЛИНА ПОДСТРОКИ
        JC      NOTFND                  ;ВЫЙТИ, ЕСЛИ СТРОКА КОРОЧЕ ПОДСТРОКИ
        INR     A                       ;СЧЕТЧИК = РАЗНИЦА В ДЛИНАХ + 1
        MOV     C,A                     ;РЕГИСТР C = СЧЕТЧИК
        SUB     A                       ;ПЕРВОНАЧАЛЬНО ИНДЕКС НАЧАЛА = 0
        STA     INDEX

        ;ИСКАТЬ ДО ТЕХ ПОР, ПОКА ОСТАТОК СТРОКИ НЕ СТАНЕТ КОРОЧЕ
        ; ПОДСТРОКИ
SLP1:
        LXI     H,INDEX                 ;УВЕЛИЧИТЬ ИНДЕКС НАЧАЛА
        INR     M
        LDA     SUBLEN                  ;B = ДЛИНА ПОДСТРОКИ
        MOV     B,A
        LHLD    SUBSTG
        XCHG                            ;DE = НАЧАЛЬНЫЙ АДРЕС ПОДСТРОКИ
        LHLD    STRING
        INX     H                       ;УВЕЛИЧИТЬ ДЛЯ СЛЕДУЮЩЕГО БАЙТА СТРОКИ
        SHLD    STRING                  ;HL = СЛЕДУЮЩИЙ АДРЕС В СТРОКЕ
                                        ;C = ТЕКУЩЕЕ ЗНАЧЕНИЕ СЧЕТЧИКА

        ;ПОПЫТАТЬСЯ ПРОВЕРИТЬ ПОДСТРОКУ НА СОВПАДЕНИЕ, НАЧИНАЯ С ИНДЕКСА
        ;ПРОВЕРКА СОСТОИТ В СРАВНЕНИИ СООТВЕТСТВУЮЩИХ СИМВОЛОВ
        ; ПО ОДНОМУ ЗА РАЗ
CMPLP:
        LDAX    D                       ;ВЗЯТЬ СИМВОЛ ПОДСТРОКИ
        CMP     M                       ;СРАВНИТЬ С СИМВОЛОМ СТРОКИ
        JNZ     SLP2                    ;ПЕРЕЙТИ, ЕСЛИ НЕ СОВПАДАЮТ
        DCR     B
        JZ      FOUND                   ;ПЕРЕЙТИ, ЕСЛИ ПОДСТРОКА НАЙДЕНА
        INX     H                       ;ПОДГОТОВИТЬ АДРЕСА ДЛЯ
        INX     D                       ; СЛЕДУЮЩЕГО СИМВОЛА
        JMP     CMPLP

        ;СЮДА ПОПАДАЕМ ПРИ НЕСОВПАДЕНИИ, ПОДСТРОКА ЕЩЕ НЕ НАЙДЕНА
SLP2:
        DCR     C
        JNZ     SLP1                    ;ЕСЛИ ПОЗВОЛЯЕТ ДЛИНА СТРОКИ,
                                        ; ПОПЫТАТЬСЯ ПРОВЕРИТЬ ДЛЯ СЛЕДУЮЩЕГО
                                        ; ИНДЕКСА
        JZ      NOTFND                  ;ИНАЧЕ ВЫЙТИ (ПОДСТРОКА НЕ НАЙДЕНА)
        ;ПОДСТРОКА НАЙДЕНА. ВЕРНУТЬСЯ С НАЧАЛЬНЫМ ИНДЕКСОМ ПОДСТРОКИ
```

```
FOUND:
        LDA     INDEX           ;ПОДСТРОКА НАЙДЕНА, A = НАЧАЛЬНЫЙ ИНДЕКС
        RET

        ;ПОДСТРОКА НЕ МОЖЕТ БЫТЬ НАЙДЕНА, ВЫЙТИ С 0 В КАЧЕСТВЕ ИНДЕКСА
NOTFND:
        SUB     A               ;ПОДСТРОКА НЕ НАЙДЕНА, A = 0
        RET

        ;ДАННЫЕ
STRING: DS      2               ;БАЗОВЫЙ АДРЕС СТРОКИ
SUBSTG: DS      2               ;БАЗОВЫЙ АДРЕС ПОДСТРОКИ
SLEN:   DS      1               ;ДЛИНА СТРОКИ
SUBLEN: DS      1               ;ДЛИНА ПОДСТРОКИ
INDEX:  DS      1               ;ТЕКУЩИЙ ИНДЕКС В СТРОКЕ


;                                                                       ;
;                                                                       ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                               ;
;                                                                       ;
;                                                                       ;

SC8C:
        LXI     H,STG           ;HL = БАЗОВЫЙ АДРЕС СТРОКИ
        LXI     D,SSTG          ;DE = БАЗОВЫЙ АДРЕС ПОДСТРОКИ
        CALL    POS             ;НАЙТИ ПОЛОЖЕНИЕ ПОДСТРОКИ
                                ; В РЕЗУЛЬТАТЕ ПОИСКА "AAB" В
                                ; "AAAAAAAAAB" ПОЛУЧАЕМ A=8

        JMP     SC8C            ;ПЕРЕЙТИ НА СЛЕДУЮЩИЙ ТЕСТ

        ;ДАННЫЕ ДЛЯ ТЕСТА, КОТОРЫЕ МОЖНО ИЗМЕНИТЬ НА ДРУГИЕ ЗНАЧЕНИЯ
STG:    DB      0AH             ;ДЛИНА СТРОКИ
        DB      'AAAAAAAAAB                     ' ;МАКС. ДЛИНА 32 БАЙТА
SSTG:   DB      3H              ;ДЛИНА ПОДСТРОКИ
        DB      'AAB                            ' ;МАКС. ДЛИНА 32 БАЙТА

        END
```

8D. КОПИРОВАНИЕ ПОДСТРОКИ ИЗ СТРОКИ (COPY)

Из строки копируется подстрока, заданная начальным индексом и числом копируемых байтов. Максимальная длина строки 255 байт, при этом действительным символам предшестует байт, содержащий длину. Если начальный индекс подстроки равен 0 (т. е. подстрока должна начинаться с байта длины) или же попадает за конец строки, то подстрока имеет длину 0, а флаг переноса устанавливается в 1. Если длина подстроки может превысить максимальную или же подстрока может выйти за конец строки, то в подстроку помещается только максимальное или допустимое число символов (до конца строки), а флаг переноса устанавливается в 1. Если же подстрока может быть сформирована в соответствии с заданием, то флаг переноса очищается.

Процедура. Если число копируемых байтов, максимальная длина подстроки или начальный индекс равны 0, то осуществляется немедленный выход

из программы. Немедленный выход осуществляется также в случае, когда начальный индекс превышает длину строки. Если эти условия отсутствуют, то проверяется, не превышает ли число копируемых байтов максимальную длину подстроки или число доступных символов в строке. Если превышает, то соответствующим образом уменьшается число копируемых байтов. Затем из строки в подстроку копируется нужное число байтов. Если подстрока может быть сформирована в соответствии с заданием, то флаг переноса очищается, если же нет, то он устанавливается в 1.

Используемые регистры: все.

Время выполнения: приблизительно 45 * ЧИСЛО КОПИРУЕМЫХ БАЙТОВ плюс дополнительно 251 такт (8080) или 46 * ЧИСЛО КОПИРУЕМЫХ БАЙТОВ плюс дополнительно 233 такта (8085). ЧИСЛО КОПИРУЕМЫХ БАЙТОВ – это или же заданное число, если при копировании не возникает проблем, или, если при копировании может произойти выход за пределы строки или подстроки, число доступных байтов или максимальная длина подстроки. Например, если ЧИСЛО КОПИРУЕМЫХ БАЙТОВ = 12_{10} ($0C_{16}$), то время выполнения будет:

45 * 12 + 251 = 540 + 251 = 791 такт (8080),
46 * 12 + 233 = 552 + 233 = 785 тактов (8085).

Размер программы: 85 байт.

Память, необходимая для данных: 2 байта в любом месте ОЗУ для максимальной длины подстроки (адрес (MAXLEN) и флага ошибки (адрес CPYERR).

Специальные случаи:

1. Если число копируемых байтов равно 0, то подстроке присваивается длина 0 и флаг переноса очищается, что указывает на отсутствие ошибок.
2. Если максимальная длина подстроки равна 0, то подстроке присваивается длина 0 и флаг переноса устанавливается в 1, что указывает на ошибку.
3. Если начальный индекс подстроки равен 0, то подстроке присваивается длина 0 и флаг переноса устанавливается в 1, что указывает на ошибку.
4. Если заданный индекс начала подстроки больше длины строки, то подстроке присваивается длина 0, флаг переноса устанавливается в 1, что указывает на ошибку.
5. Если подстрока может выйти за конец исходной строки, то в подстроку помещаются все доступные символы и флаг переноса устанавливается в 1, что указывает на ошибку.
6. Если подстрока может превысить заданную максимальную длину, то в подстроку помещается только заданное максимальное число символов. Флаг переноса устанавливается в 1, что указывает на ошибку.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес подстроки в регистрах D и E.
Базовый адрес строки в регистрах H и L.
Количество копируемых байтов в регистре B.
Начальный индекс, с которого следует начинать копирование, в регистре C.
Максимальная длина подстроки в регистре A.

УСЛОВИЯ НА ВЫХОДЕ

Подстрока содержит символы, скопированные из строки. Если начальный индекс подстроки равен 0, максимальная длина подстроки равна 0 или на-

чальный индекс указывает за пределы строки, то подстрока будет иметь длину 0, а флаг переноса установлен в 1. Если подстрока простирается за конец строки или же может превысить заданную максимальную длину, то в подстроку копируется только доступное число символов строки (до максимальной длины подстроки); кроме того, в этом случае устанавливается флаг переноса. Если при формировании подстроки не возникает каких-либо проблем, флаг переноса очищается.

ПРИМЕРЫ

1. Данные: строка = 10´LET Y1 = R7 + X4´ ($10_{16} = 16_{10}$ – длина строки),
максимальная длина подстроки = 2,
число копируемых байтов = 2,
начальный индекс = 5.
Результат: подстрока = 02 Y1 (2 – длина подстроки),
были скопированы из строки два байта, начиная с 5-го символа (т. е. символы 5 и 6),
флаг переноса = 0, так как при формировании подстроки не возникло проблем.
2. Данные: строка = 0E´8657 POWELL ST´ ($0E_{16} = 14_{10}$ – длина строки),
максимальная длина подстроки = $10_{16} = 16_{10}$,
число копируемых байтов = $0D_{16} = 13_{10}$,
начальный индекс = 6,
Результат: подстрока = 09´POWELL ST´ (09 – длина подстроки),
флаг переноса = 1, так как в строке было недостаточно символов для получения копии заданной длины.
3. Данные: строка = 16´9414 HEGENBERGER DRIVE´ ($16_{16} = 22_{10}$ – длина строки),
максимальная длина подстроки = $10_{16} = 16_{10}$,
число копируемых байтов = $11_{16} = 17_{10}$,
начальный индекс = 6.
Результат: подстрока = 10´HEGENBERGER DRIV´ ($10_{16} = 16_{10}$ – длина подстроки),
флаг переноса = 1, так как число копируемых байтов превышает максимальную длину подстроки.

```
;                                                                    ;
;                                                                    ;
;                                                                    ;
;                                                                    ;
;       ЗАГОЛОВОК:      КОПИРОВАНИЕ ПОДСТРОКИ ИЗ СТРОКИ              ;
;       ИМЯ:            COPY                                         ;
;                                                                    ;
;                                                                    ;


;                                                                    ;
;       НАЗНАЧЕНИЕ:     КОПИРУЕТ ПОДСТРОКУ, ЗАДАННУЮ НАЧАЛЬНЫМ ИНДЕКСОМ ;
;                       И ДЛИНОЙ В БАЙТАХ, ИЗ СТРОКИ                 ;
;                                                                    ;
;       ВХОД:           ПАРА РЕГИСТРОВ H = АДРЕС ИСХОДНОЙ СТРОКИ     ;
;                       ПАРА РЕГИСТРОВ D = АДРЕС СТРОКИ НАЗНАЧЕНИЯ   ;
;                       РЕГИСТР A = МАКСИМАЛЬНАЯ ДЛИНА СТРОКИ НАЗНАЧЕНИЯ;
;                       РЕГИСТР B = КОЛИЧЕСТВО БАЙТОВ, КОТОРЫЕ ДОЛЖНЫ ;
;                                   БЫТЬ СКОПИРОВАНЫ                 ;
;                       РЕГИСТР C = НАЧАЛЬНЫЙ ИНДЕКС В ИСХОДНОЙ СТРОКЕ. ;
;                                   ИНДЕКС 1 ЯВЛЯЕТСЯ ИНДЕКСОМ ПЕРВОГО ;
;                                   СИМВОЛА В СТРОКЕ                 ;
```

```
;                                                                           ;
;                           МАКСИМАЛЬНАЯ ДЛИНА СТРОКИ 255 БАЙТ ПЛЮС ОДИН    ;
;                           ПРЕДШЕСТВУЮЩИЙ БАЙТ С ДЛИНОЙ СТРОКИ             ;
;                                                                           ;
;       ВЫХОД:            СТРОКА НАЗНАЧЕНИЯ := ПОДСТРОКА ИЗ СТРОКИ          ;
;                         ЕСЛИ НЕТ ОШИБОК, ТО                               ;
;                           ФЛАГ ПЕРЕНОСА := 0                              ;
;                         ИНАЧЕ                                             ;
;                           СЛЕДУЮЩИЕ УСЛОВИЯ ВЫЗЫВАЮТ ОШИБКУ И УСТАНОВКУ   ;
;                           ФЛАГА ПЕРЕНОСА В 1:                             ;
;                           ЕСЛИ ИНДЕКС = 0, ИЛИ МАКСИМАЛЬНАЯ ДЛИНА = 0,    ;
;                             ИЛИ ИНДЕКС > ДЛИНЫ ИСХОДНОЙ СТРОКИ,ТО ДЛИНА   ;
;                             СТРОКИ НАЗНАЧЕНИЯ БУДЕТ РАВНА 0.              ;
;                         ЕСЛИ (ИНДЕКС + ЧИСЛО КОПИРУЕМЫХ БАЙТОВ - 1) >     ;
;                           ДЛИНЫ ИСХОДНОЙ СТРОКИ, ТО                       ;
;                           ВСЯ ИСХОДНАЯ СТРОКА ПЕРЕПИСЫВАЕТСЯ В СТРОКУ     ;
;                           НАЗНАЧЕНИЯ                                      ;
;                                                                           ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ                                          ;
;                                                                           ;
;       ВРЕМЯ:             ПРИБЛИЗИТЕЛЬНО (45 * ЧИСЛО КОПИРУЕМЫХ БАЙТОВ)    ;
;                          ТАКТОВ ПЛЮС 251 ТАКТ ДЛЯ 8080                    ;
;                          ПРИБЛИЗИТЕЛЬНО (46 * ЧИСЛО КОПИРУЕМЫХ БАЙТОВ)    ;
;                          ТАКТОВ ПЛЮС 233 ТАКТА ДЛЯ 8085                   ;
;                                                                           ;
;       РАЗМЕР:           ПРОГРАММА - 85 БАЙТ                               ;
;                         ДАННЫЕ    -  2 БАЙТА                              ;
;                                                                           ;
;                                                                           ;

COPY:
        ;СОХРАНИТЬ МАКСИМАЛЬНУЮ ДЛИНУ СТРОКИ НАЗНАЧЕНИЯ
        STA     MAXLEN          ;СОХРАНИТЬ МАКСИМАЛЬНУЮ ДЛИНУ

        ;ЗАДАТЬ НАЧАЛЬНЫЕ ЗНАЧЕНИЯ СТРОКИ НАЗНАЧЕНИЯ И ФЛАГА ОШИБКИ
        SUB     A
        STAX    D               ;ДЛИНА СТРОКИ НАЗНАЧЕНИЯ РАВНА 0
        STA     CPYERR          ;СНАЧАЛА СЧИТАЕТСЯ, ЧТО ОШИБОК НЕТ

        ;ЕСЛИ ЧИСЛО КОПИРУЕМЫХ БАЙТОВ РАВНО 0, ТО ВЫЙТИ БЕЗ ОШИБКИ
        ORA     B               ;ПРОВЕРИТЬ ЧИСЛО КОПИРУЕМЫХ БАЙТОВ
        RZ                      ; ВЫЙТИ БЕЗ ОШИБКИ
                                ; ФЛАГ ПЕРЕНОСА = 0

        ;ЕСЛИ МАКСИМАЛЬНАЯ ДЛИНА РАВНА 0, ТО ВЫЙТИ ПО ОШИБКЕ
        LDA     MAXLEN          ;ПРОВЕРИТЬ МАКСИМАЛЬНУЮ ДЛИНУ ПОДСТРОКИ
        ORA     A
        JZ      EREXIT          ;ВЫЙТИ ПО ОШИБКЕ, ЕСЛИ МАКСИМАЛЬНАЯ
                                ; ДЛИНА РАВНА 0

        ;ЕСЛИ НАЧАЛЬНЫЙ ИНДЕКС РАВЕН 0, ТО ВЫЙТИ ПО ОШИБКЕ
        MOV     A,C             ;ПРОВЕРИТЬ НАЧАЛЬНЫЙ ИНДЕКС
        ORA     A
        JZ      EREXIT          ;ВЫЙТИ ПО ОШИБКЕ, ЕСЛИ ИНДЕКС РАВЕН 0

        ;ЕСЛИ НАЧАЛЬНЫЙ ИНДЕКС БОЛЬШЕ ДЛИНЫ ИСХОДНОЙ СТРОКИ, ТО
        ; ВЫЙТИ ПО ОШИБКЕ
```

```
        MOV     A,M                 ;ВЗЯТЬ ДЛИНУ ИСХОДНОЙ СТРОКИ
        CMP     C                   ;СРАВНИТЬ С НАЧАЛЬНЫМ ИНДЕКСОМ
        RC                          ;ВЫЙТИ ПО ОШИБКЕ, ЕСЛИ ДЛИНА МЕНЬШЕ
                                    ; ИНДЕКСА, ФЛАГ ПЕРЕНОСА = 1

        ;ПРОВЕРИТЬ, ПОПАДАЕТ ЛИ КОПИРУЕМАЯ ОБЛАСТЬ В ИСХОДНУЮ СТРОКУ
        ;ЕСЛИ НЕТ, ТО СКОПИРОВАТЬ ТОЛЬКО КОНЕЦ СТРОКИ
        ;КОПИРУЕМАЯ ОБЛАСТЬ ПОПАДАЕТ В ИСХОДНУЮ СТРОКУ, ЕСЛИ
        ; ИНДЕКС + ЧИСЛО КОПИРУЕМЫХ СИМВОЛОВ - 1 МЕНЬШЕ ИЛИ РАВНО
        ; ДЛИНЕ ИСХОДНОЙ СТРОКИ
        ;ЗАМЕТИМ, ЧТО ДЛИНА СТРОК НЕ МОЖЕТ БЫТЬ БОЛЬШЕ 255 БАЙТ
        MOV     A,C                 ;СФОРМИРОВАТЬ НАЧАЛЬНЫЙ ИНДЕКС +
                                    ; ДЛИНА КОПИИ
        ADD     B
        JC      RECALC              ;ПЕРЕЙТИ, ЕСЛИ СУММА > 255
        DCR     A
        CMP     M                   ;СРАВНИТЬ СУММУ С ДЛИНОЙ СТРОКИ
        JC      CNT1OK              ;ПЕРЕЙТИ, ЕСЛИ СУММА БОЛЬШЕ,
                                    ; ЧЕМ НУЖНО ДЛЯ КОПИИ
        JZ      CNT1OK              ;ПЕРЕЙТИ, ЕСЛИ РОВНО СТОЛЬКО, СКОЛЬКО
                                    ; НУЖНО

        ;ВЫЗЫВАЮЩАЯ ПРОГРАММА ЗАПРОСИЛА СЛИШКОМ МНОГО СИМВОЛОВ.
        ; ВЕРНУТЬ ВСЕ МЕЖДУ ИНДЕКСОМ И КОНЦОМ ИСХОДНОЙ СТРОКИ.
        ; СЧЕТЧИК := ДЛИНА ИСХОДНОЙ СТРОКИ - ИНДЕКС + 1
RECALC:
        MVI     A,0FFH              ;УСТАНОВИТЬ ФЛАГ УСЕЧЕНИЯ СЧЕТЧИКА
        STA     CPYERR
        MOV     A,M                 ;СЧЕТЧИК = ДЛИНА - ИНДЕКС + 1
        SUB     C
        INR     A
        MOV     B,A                 ;ИЗМЕНИТЬ ЧИСЛО КОПИРУЕМЫХ БАЙТОВ

        ;ПРОВЕРИТЬ, МЕНЬШЕ ЛИ СЧЕТЧИК МАКСИМАЛЬНОЙ ДЛИНЫ СТРОКИ
        ; НАЗНАЧЕНИЯ ИЛИ РАВЕН ЕЙ. ЕСЛИ НЕТ, УСТАНОВИТЬ СЧЕТЧИК РАВНЫМ
        ; МАКСИМАЛЬНОЙ ДЛИНЕ
        ;ЕСЛИ СЧЕТЧИК > МАКСИМАЛЬНОЙ ДЛИНЫ, ТО
        ; СЧЕТЧИК := МАКСИМАЛЬНАЯ ДЛИНА
CNT1OK:
        LDA     MAXLEN              ;МАКСИМАЛЬНАЯ ДЛИНА ДОСТАТОЧНО БОЛЬШАЯ?
        CMP     B
        JNC     CNT2OK              ;ПЕРЕЙТИ, ЕСЛИ ДА
        MOV     B,A                 ;ИНАЧЕ ОГАНИЧИТЬ КОПИЮ ДО МАКСИМАЛЬНОЙ
                                    ; ДЛИНЫ
        MVI     A,0FFH              ;УСТАНОВИТЬ ФЛАГ ПЕРЕПОЛНЕНИЯ
        STA     CPYERR

        ;ПЕРЕСЛАТЬ ПОДСТРОКУ В СТРОКУ НАЗНАЧЕНИЯ
CNT2OK:
        MOV     A,B                 ;ПРОВЕРИТЬ ЧИСЛО КОПИРУЕМЫХ БАЙТОВ
        ORA     A
        JZ      EREXIT              ;ВЫЙТИ ПО ОШИБКЕ, ЕСЛИ НЕЧЕГО КОПИРОВАТЬ
        MVI     B,0                 ;НАЧАТЬ КОПИРОВАНИЕ С НАЧАЛЬНОГО ИНДЕКСА
        DAD     B
        STAX    D                   ;УСТАНОВИТЬ ДЛИНУ СТРОКИ НАЗНАЧЕНИЯ
        MOV     B,A                 ;ВОССТАНОВИТЬ ЧИСЛО БАЙТОВ
```

```
COPYLP:
        MOV     A,M             ;ВЗЯТЬ СИМВОЛ ИЗ СТРОКИ
        INX     H               ;УВЕЛИЧИТЬ АДРЕС ДЛЯ СЛЕДУЮЩЕГО БАЙТА
                                ; В ИСХОДНОЙ СТРОКЕ
        INX     D               ;УВЕЛИЧИТЬ АДРЕС ДЛЯ СЛЕДУЮЩЕГО БАЙТА
                                ; В СТРОКЕ НАЗНАЧЕНИЯ
        STAX    D               ;ЗАПОМНИТЬ СИМВОЛ В ПОДСТРОКЕ
        DCR     B
        JNZ     COPYLP          ;ПРОДОЛЖАТЬ ДО ОКОНЧАНИЯ РАБОТЫ

        ;ПРОВЕРИТЬ НА ОШИБКУ КОПИИ
        LDA     CPYERR          ;ПРОВЕРИТЬ НА ОШИБКУ
OKEXIT:
        ORA     A
        RZ                      ;ЕСЛИ НЕТ ОШИБОК, ВЕРНУТЬСЯ
                                ; С ФЛАГОМ ПЕРЕНОСА = 0

        ;ВЫХОД ПО ОШИБКЕ
EREXIT:
        STC                     ;ПРИ ОШИБКЕ УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА
        RET

        ;СЕКЦИЯ ДАННЫХ
MAXLEN: DS      1               ;МАКСИМАЛЬНАЯ ДЛИНА СТРОКИ НАЗНАЧЕНИЯ
CPYERR: DS      1               ;ФЛАГ ОШИБКИ КОПИИ


;                                                                       ;
;                                                                       ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                               ;
;                                                                       ;
;                                                                       ;

SCBD:
        LXI     H,SSTG          ;БАЗОВЫЙ АДРЕС ИСХОДНОЙ СТРОКИ
        LXI     D,DSTG          ;БАЗОВЫЙ АДРЕС СТРОКИ НАЗНАЧЕНИЯ
        LDA     IDX
        MOV     C,A             ;НАЧАЛЬНЫЙ ИНДЕКС ДЛЯ КОПИРОВАНИЯ
        LDA     CNT
        MOV     B,A             ;ЧИСЛО КОПИРУЕМЫХ БАЙТОВ
        LDA     MXLEN           ;МАКСИМАЛЬНАЯ ДЛИНА ПОДСТРОКИ
        CALL    COPY            ;КОПИРОВАТЬ ПОДСТРОКУ
                                ;КОПИРОВАНИЕ 3 СИМВОЛОВ, НАЧИНАЯ С
                                ; ИНДЕКСА 4 ИЗ '12.345E+10' ДАЕТ '345'

        JMP     SCBD            ;ЦИКЛ ДЛЯ СЛЕДУЮЩЕЙ ПРОВЕРКИ

        ;СЕКЦИЯ ДАННЫХ
IDX:    DB      4               ;НАЧАЛЬНЫЙ ИНДЕКС ДЛЯ КОПИРОВАНИЯ
CNT:    DB      3               ;ЧИСЛО КОПИРУЕМЫХ СИМВОЛОВ
MXLEN:  DB      20H             ;МАКСИМАЛЬНАЯ ДЛИНА СТРОКИ НАЗНАЧЕНИЯ
SSTG:   DB      0AH             ;ДЛИНА СТРОКИ
        DB      '12.345E+10                     ' ;МАКС. ДЛИНА 32 БАЙТА
DSTG:   DB      0               ;ДЛИНА ПОДСТРОКИ
        DB      '                               ' ;МАКС. ДЛИНА 32 БАЙТА

        END
```

Из строки удаляется подстрока, заданная начальным индексом и длиной. Максимальная длина строки 256 байт, при этом действительным символам предшествует байт, содержащий длину. Если удаление может быть выполнено в соответствии с заданием, флаг переноса очищается. Если начальный индекс равен 0 или превышает длину строки, флаг переноса устанавливается в 1; в любом случае строка остается без изменения. Если при удалении происходит выход за конец строки, флаг переноса устанавливается в 1 и удаляются только символы, начинающиеся с начального индекса, до конца строки.

Процедура. Если начальный индекс равен 0 или число удаляемых байтов равно 0, то осуществляется немедленный выход из программы. Выход происходит также в том случае, когда начальный индекс попадает за пределы строки. Если эти условия отсутствуют, то проверяется, не простирается ли строка за пределы удаляемой области. Если этого не происходит, то просто усекается строка – устанавливается новая длина в начальный индекс минус 1. Если строка выходит за пределы удаляемой области, то результирующая строка уплотняется перемещением вниз байтов, лежащих за удаляемой областью. Затем определяется новая длина строки и осуществляется выход с флагом переноса, равным 0, если было удалено заданное число символов, или устанавливается флаг переноса в 1, если были какие-либо ошибки.

Используемые регистры: все.

Время выполнения: приблизительно 39 * ЧИСЛО ПЕРЕСЛАННЫХ ВНИЗ БАЙТОВ + 237 тактов (8080) или 40 * ЧИСЛО ПЕРЕСЛАННЫХ ВНИЗ БАЙТОВ + + 226 тактов (8085), где число ПЕРЕСЛАННЫХ ВНИЗ БАЙТОВ равно нулю, если строка была усечена, или равно ДЛИНА СТРОКИ – НАЧАЛЬНЫЙ ИНДЕКС – – ЧИСЛО УДАЛЯЕМЫХ БАЙТОВ + 1, если строка уплотнялась. Таким образом, в случае, когда при удалении в строке образуется "дыра", которая должна быть заполнена с помощью уплотнения, требуется дополнительное время.

Примеры

1. ДЛИНА СТРОКИ = 20_{16} (32_{10}),
   НАЧАЛЬНЫЙ ИНДЕКС = 19_{16} (25_{10}),
   ЧИСЛО УДАЛЯЕМЫХ БАЙТОВ = 08.

Так как в строке, начиная с индекса 19_{16}, остается ровно 8 байт, то единственное, что должна сделать подпрограмма, так это усечь строку (т. е. обрезать конец строки). Это требует:

39 * 0 + 237 = 237 тактов (8080),
40 * 0 + 226 = 226 тактов (8085).

2. ДЛИНА СТРОКИ = 40_{16} (64_{10}),
   НАЧАЛЬНЫЙ ИНДЕКС = 19_{16} (25_{10}),
   ЧИСЛО УДАЛЯЕМЫХ БАЙТОВ = 08.

Так как за удаляемой областью есть 20_{16} (32_{10}) байт, подпрограмма должна переслать их вниз на восемь позиций для заполнения "дыры". Таким образом, ЧИСЛО ПЕРЕСЫЛАЕМЫХ ВНИЗ БАЙТОВ = 32_{10}, и время выполнения будет:

39 * 32 + 237 = 1248 + 237 = 1485 тактов (8080),
40 * 32 + 226 = 1280 + 226 = 1506 тактов (8085).

Размер программы: 68 байт.

Память, необходимая для данных: 1 байт в любом месте ОЗУ для флага ошибки (адрес DELERR).

Специальные случаи:

1. Если число удаляемых байтов равно 0, то осуществляется выход с очищенным флагом переноса (нет ошибок), строка при этом остается без изменения.

2. Если начальный индекс попадает за пределы строки, то осуществляется выход с флагом переноса, установленным в 1 (указывает на ошибку), при этом строка остается без изменения.

3. Если число удаляемых байтов превышает возможное число, то удаляются все байты от начального индекса до конца строки и осуществляется выход с флагом переноса, установленным в 1 (указывает на ошибку).

УСЛОВИЯ НА ВХОДЕ

Базовый адрес строки в регистрах H и L.
Число удаляемых байтов в регистре B.
Начальный индекс, с которого необходимо удалить байты, в регистре C.

УСЛОВИЯ НА ВЫХОДЕ

Подстрока удаляется из строки. Если не возникает ошибок, флаг переноса очищается. Если начальный индекс равен 0 или попадает за пределы строки, флаг переноса устанавливается в 1, а строка остается без изменения. Если число удаляемых байтов выходит за конец строки, то флаг переноса устанавливается в 1, а символы с начального индекса до конца строки удаляются.

ПРИМЕРЫ

1. Данные: строка = 26´SALES FOR MARCH AND APRIL OF THIS YEAR´
($26_{16} = 38_{10}$ – длина строки),
число удаляемых символов = $0A_{16} = 10_{10}$,
начальный индекс для удаления = $10_{16} = 16_{10}$.
Результат: строка = 1C´SALES FOR MARCH OF THIS YEAR´
($1C_{16} = 28_{10}$ – длина строки после удаления десяти байтов, начиная с 16-го символа; удалены символы ´AND APRIL´),
флаг переноса = 0, так как при удалении не возникло проблем.

2. Данные: строка = 28´THE PRICE IS $ 3.00 ($ 2.00 BEFORE JUNE 1)´
($28_{16} = 40_{10}$ – длина строки),
число удаляемых байтов = $30_{16} = 48_{10}$,
начальный индекс для удаления = $13_{16} = 19_{10}$.
Результат: строка = 12´THE PRICE IS $3.00´
($12_{16} = 18_{10}$ – длина строки после удаления всех остальных байтов),
флаг переноса = 1, так как в строке не было стольких байтов, сколько необходимо было удалить.

```
;                                                            ;
;                                                            ;
;                                                            ;
;                                                            ;
;       ЗАГОЛОВОК:      УДАЛЕНИЕ ПОДСТРОКИ ИЗ СТРОКИ         ;
;       ИМЯ:            DELETE                               ;
;                                                            ;
;                                                            ;

;                                                            ;
```

```
;       НАЗНАЧЕНИЕ:        УДАЛЯЕТ ПОДСТРОКУ, ЗАДАННУЮ НАЧАЛЬНЫМ ИНДЕКСОМ  ;
;                          И ДЛИНОЙ В БАЙТАХ, ИЗ СТРОКИ                    ;
;                                                                          ;
;       ВХОД:              ПАРА РЕГИСТРОВ H = БАЗОВЫЙ АДРЕС СТРОКИ         ;
;                          РЕГИСТР B = ЧИСЛО БАЙТОВ, КОТОРЫЕ ДОЛЖНЫ БЫТЬ   ;
;                                      УДАЛЕНЫ                             ;
;                          РЕГИСТР C = НАЧАЛЬНЫЙ ИНДЕКС В СТРОКЕ. ИНДЕКС 1 ;
;                                      ЯВЛЯЕТСЯ ИНДЕКСОМ ПЕРВОГО СИМВОЛА   ;
;                                      В СТРОКЕ                            ;
;                                                                          ;
;                             МАКСИМАЛЬНАЯ ДЛИНА СТРОКИ 255 БАЙТ ПЛЮС ОДИН ;
;                             ПРЕДШЕСТВУЮЩИЙ БАЙТ С ДЛИНОЙ СТРОКИ          ;
;                                                                          ;
;       ВЫХОД:             ПОДСТРОКА УДАЛЯЕТСЯ                             ;
;                          ЕСЛИ НЕТ ОШИБОК, ТО                             ;
;                            ФЛАГ ПЕРЕНОСА := 0                            ;
;                          ИНАЧЕ                                           ;
;                            СЛЕДУЮЩИЕ УСЛОВИЯ ВЫЗЫВАЮТ ОШИБКУ И УСТАНОВКУ ;
;                            ФЛАГА ПЕРЕНОСА В 1:                           ;
;                            ЕСЛИ ИНДЕКС = 0 ИЛИ ИНДЕКС > ДЛИНЫ СТРОКИ,    ;
;                              ТО СТРОКА НЕ ИЗМЕНЯЕТСЯ                     ;
;                            ЕСЛИ КОЛИЧЕСТВО УДАЛЯЕМЫХ БАЙТОВ СЛИШКОМ      ;
;                              ВЕЛИКО, ТО УДАЛЯЮТСЯ ТОЛЬКО СИМВОЛЫ, НАЧИНАЯ;
;                              С ИНДЕКСА И ДО КОНЦА СТРОКИ                 ;
;                                                                          ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ                                         ;
;                                                                          ;
;       ВРЕМЯ:             ПРИБЛИЗИТЕЛЬНО 39 * (ДЛИНА СТРОКИ - ИНДЕКС -    ;
;                          ЧИСЛО УДАЛЯЕМЫХ СИМВОЛОВ + 1) ПЛЮС 237 ТАКТОВ   ;
;                          ДЛЯ 8080                                        ;
;                          ПРИБЛИЗИТЕЛЬНО 40 * (ДЛИНА СТРОКИ - ИНДЕКС -    ;
;                          ЧИСЛО УДАЛЯЕМЫХ СИМВОЛОВ + 1) ПЛЮС 226 ТАКТОВ   ;
;                          ДЛЯ 8080                                        ;
;                                                                          ;
;       РАЗМЕР:            ПРОГРАММА - 68 БАЙТ                             ;
;                          ДАННЫЕ    -  1 БАЙТ                             ;
;                                                                          ;
;                                                                          ;

DELETE:
        ;ЗАДАТЬ НАЧАЛЬНОЕ ЗНАЧЕНИЕ ИНДИКАТОРА ОШИБКИ (DELERR) РАВНЫМ 0
        SUB     A
        STA     DELERR          ;СЧИТАТЬ, ЧТО ОШИБОК НЕТ

        ;ПРОВЕРИТЬ ИНДЕКС И СЧЕТЧИК НА НЕРАВЕНСТВО НУЛЮ
        ORA     B               ;ПРОВЕРИТЬ ЧИСЛО УДАЛЯЕМЫХ БАЙТОВ
        RZ                      ;ЕСЛИ УДАЛЯЕТСЯ 0 БАЙТ, ВОЗВРАТИТЬСЯ
                                ; С ФЛАГОМ ПЕРЕНОСА = 0 (НЕТ ОШИБОК)
        MOV     A,C             ;ПРОВЕРИТЬ НАЧАЛЬНЫЙ ИНДЕКС
        ORA     A
        STC
        RZ                      ;ЕСЛИ НАЧАЛЬНЫЙ ИНДЕКС = 0,
                                ; ТО ВЫЙТИ ПО ОШИБКЕ (ФЛАГ ПЕРЕНОСА = 1)

        ;ПРОВЕРИТЬ, НАХОДИТСЯ ЛИ НАЧАЛЬНЫЙ ИНДЕКС ВНУТРИ СТРОКИ.
        ; ЕСЛИ НЕТ, ТО ОШИБКА
        MOV     A,M             ;ВЗЯТЬ ДЛИНУ
```

```
        CMP     C               ;ИНДЕКС ВНУТРИ СТРОКИ?
        RC                      ;НЕТ, ВЫЙТИ ПО ОШИБКЕ

        ;УБЕДИТЬСЯ, ЧТО ЕСТЬ ДОСТАТОЧНО СИМВОЛОВ ДЛЯ УДАЛЕНИЯ.
        ; ЕСЛИ НЕТ, УДАЛИТЬ ТОЛЬКО СИМВОЛЫ, НАЧИНАЯ С ИНДЕКСА И ДО
        ; КОНЦА СТРОКИ.
        ; ЕСЛИ ИНДЕКС + ЧИСЛО СИМВОЛОВ - 1 > ДЛИНЫ СТРОКИ, ТО
        ;   ЧИСЛО СИМВОЛОВ := ДЛИНА СТРОКИ - ИНДЕКС + 1
        MOV     A,C             ;ВЗЯТЬ ИНДЕКС
        ADD     B               ;ПРИБАВИТЬ ЧИСЛО УДАЛЯЕМЫХ СИМВОЛОВ
        JC      TRUNC           ;ОБРЕЗАТЬ СТРОКУ, ЕСЛИ СУММА > 255
        MOV     E,A             ;СОХРАНИТЬ СУММУ В КАЧЕСТВЕ НАЧАЛЬНОГО
                                ; ИНДЕКСА ДЛЯ УДАЛЕНИЯ
        DCR     A
        CMP     M               ;СРАВНИТЬ С ДЛИНОЙ
        JC      CNTOK           ;ПЕРЕЙТИ, ЕСЛИ БОЛЬШЕ, ЧЕМ ЕСТЬ СИМВОЛОВ
                                ; ДЛЯ УДАЛЕНИЯ
        JZ      TRUNC           ;ОБРЕЗАТЬ, НО НЕ УКАЗЫВАТЬ НА ОШИБКУ
                                ; (СИМВОЛОВ РОВНО СТОЛЬКО, СКОЛЬКО НАДО)
        MVI     A,0FFH
        STA     DELERR          ;УСТАНОВИТЬ ФЛАГ ОШИБКИ - ДЛЯ УДАЛЕНИЯ
                                ; НЕДОСТАТОЧНО СИМВОЛОВ

        ;ОБРЕЗАТЬ СТРОКУ - НЕТ ДОСТАТОЧНОГО ЧИСЛА СИМВОЛОВ.
        ; ДЛИНА СТРОКИ - ИНДЕКС - 1
TRUNC:
        MOV     A,C             ;ДЛИНА СТРОКИ = ИНДЕКС - 1
        DCR     A
        MOV     M,A
        LDA     DELERR
        RAR                     ;ЕСЛИ НЕТ ОШИБОК, ТО ФЛАГ ПЕРЕНОСА = 0
        RET                     ;ВЫЙТИ ИЗ ПОДПРОГРАММЫ

        ;УДАЛИТЬ ПОДСТРОКУ, ТАК КАК СИМВОЛОВ ДОСТАТОЧНО.
        ; ПЕРЕСЛАТЬ ВСЕ СИМВОЛЫ, ЛЕЖАЩИЕ ВЫШЕ УДАЛЯЕМОЙ ОБЛАСТИ, ВНИЗ
        ;НОВАЯ ДЛИНА = ДЛИНА - ЧИСЛО УДАЛЯЕМЫХ ИЗ СТАРОЙ СТРОКИ
        ; БАЙТОВ
CNTOK:
        MOV     A,M
        MOV     D,A             ;СОХРАНИТЬ СТАРУЮ ДЛИНУ
        SUB     B               ;УСТАНОВИТЬ НОВУЮ ДЛИНУ
        MOV     M,A

        ;ВЫЧИСЛИТЬ ЧИСЛО ПЕРЕСЫЛАЕМЫХ СИМВОЛОВ
        ; ЧИСЛО ПЕРЕСЫЛАЕМЫХ СИМВОЛОВ = ДЛИНА СТРОКИ - (ИНДЕКС +
        ;   ЧИСЛО БАЙТОВ) + 1
        MOV     A,D             ;ВЗЯТЬ СТАРУЮ ДЛИНУ
        SUB     E               ;ВЫЧЕСТЬ ИНДЕКС + ЧИСЛО БАЙТОВ
        INR     A               ;A = ЧИСЛО ПЕРЕСЫЛАЕМЫХ СИМВОЛОВ

        ;ВЫЧИСЛИТЬ АДРЕСА ИСХОДНОЙ ОБЛАСТИ И ОБЛАСТИ НАЗНАЧЕНИЯ
        ; ДЛЯ ПЕРЕСЫЛКИ
        ;ИСХОДНЫЙ АДРЕС = БАЗА + ИНДЕКС + ЧИСЛО УДАЛЯЕМЫХ СИМВОЛОВ
        ;АДРЕС НАЗНАЧЕНИЯ = БАЗА + ИНДЕКС
        PUSH    H               ;СОХРАНИТЬ АДРЕС СТРОКИ
        MVI     B,0             ;АДРЕС НАЗНАЧЕНИЯ = БАЗА + ИНДЕКС
        DAD     B
```

```
        XTHL                            ;ИСХОДНЫЙ АДРЕС = БАЗА + ИНДЕКС +
        MVI     D,0                     ; ЧИСЛО УДАЛАЯЕМЫХ БАЙТОВ
        DAD     D                       ;HL = ИСХОДНЫЙ АДРЕС (ВЫШЕ УДАЛЯЕМОЙ
                                        ; ОБЛАСТИ
        POP     D                       ;DE = АДРЕС НАЗНАЧЕНИЯ
        MOV     B,A                     ;B = ЧИСЛО ПЕРЕСЫЛАЕМЫХ СИМВОЛОВ

MVLP:
        MOV     A,M                     ;ВЗЯТЬ СИМВОЛ
        STAX    D                       ;ПЕРЕСЛАТЬ ЕГО ВНИЗ
        INX     H                       ;ПРОДОЛЖИТЬ ДЛЯ СЛЕДУЮЩЕГО СИМВОЛА
        INX     D
        DCR     B
        JNZ     MVLP

        ;ВЫХОД БЕЗ ОШИБКИ
OKEXIT:
        ORA     A                       ;ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА, НЕТ ОШИБОК
        RET

        ;ДАННЫЕ
DELERR: DS      1                       ;ФЛАГ ОШИБКИ ПРИ УДАЛЕНИИ


;                                                                       ;
;                                                                       ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                               ;
;                                                                       ;
;                                                                       ;

SC8E:
        LXI     H,SSTG  ;HL = БАЗОВЫЙ АДРЕС СТРОКИ
        LDA     IDX
        MOV     C,A     ;C = НАЧАЛЬНЫЙ ИНДЕКС ДЛЯ УДАЛЕНИЯ
        LDA     CNT
        MOV     B,A     ;B = ЧИСЛО УДАЛЯЕМЫХ СИМВОЛОВ
        CALL    DELETE  ;УДАЛИТЬ СИМВОЛЫ
                        ;ПОСЛЕ УДАЛЕНИЯ ИЗ "JOE HANDOVER" 4 СИМВОЛОВ,
                        ; НАЧИНАЯ С ИНДЕКСА 1, ОСТАЕТСЯ "HANDOVER"

        JMP     SC8E    ;ЦИКЛ ДЛЯ СЛЕДУЮЩЕГО ТЕСТА

        ;СЕКЦИЯ ДАННЫХ
IDX:    DB      1                       ;НАЧАЛЬНЫЙ ИНДЕКС ДЛЯ УДАЛЕНИЯ
CNT:    DB      4                       ;ЧИСЛО УДАЛЯЕМЫХ СИМВОЛОВ
SSTG:   DB      12                      ;ДЛИНА СТРОКИ
        DB      'JOE HANDOVER'

        END
```

8F. ВСТАВКА ПОДСТРОКИ В СТРОКУ (INSERT)

Вставляет подстроку в строку по заданному начальному индексу. Максимальная длина как строки, так и подстроки равна 255 байт, при этом действительным символам предшествует байт, содержащий длину. Если при выполнении вставки не возникло проблем, то флаг переноса очищается. Если

начальный индекс равен 0 или выходит за пределы строки, то флаг переноса устанавливается в 1. Во втором случае (когда индекс выходит за пределы строки) подстрока присоединяется к концу строки. Флаг переноса устанавливается также в том случае, если длина строки после вставки превышает заданную максимальную; при этом вставляется только часть подстроки, доводящая длину строки до максимальной.

Процедура. Если начальный индекс или длина подстроки равны 0, то происходит немедленный выход из программы. Если они не равны 0, то проверяется, не превысит ли в результате вставки длина строки максимальную. Если это может произойти, подстрока усекается. Затем проверяется, попадает ли начальный индекс в строку. Если нет, то подстрока просто присоединяется с помощью пересылки ее в ячейки памяти, следующие сразу за концом строки. Если начальный индекс попадает внутрь строки, то сначала должно освободиться место для вставки с помощью пересылки оставшихся символов в большие адреса памяти. Во избежание записи поверх данных эта пересылка должна начинаться с больших адресов. Окончательно подстрока пересылается в освободившуюся область. Затем определяется новая длина строки и осуществляется выход с соответствующим образом установленным флагом переноса (0, если не было проблем, или 1, если начальный индекс был равен 0, подстрока была усечена, или начальный индекс попал за пределы строки).

Используемые регистры: все.

Время выполнении: приблизительно 39 * ЧИСЛО ПЕРЕСЫЛАЕМЫХ БАЙТОВ + + 39 * ЧИСЛО ВСТАВЛЯЕМЫХ БАЙТОВ + 300 (8080) или 40 * ЧИСЛО ПЕРЕСЫЛАЕМЫХ БАЙТОВ + 40 * ЧИСЛО ВСТАВЛЯЕМЫХ БАЙТОВ + 288 (8085). ЧИСЛО ПЕРЕСЫЛАЕМЫХ БАЙТОВ – число байтов, которое необходимо переслать при освобождении области для вставки. Если начальный индекс выходит за конец строки, это число равно 0, так как подстрока просто присоединяется к строке. В противном случае это число равно ДЛИНА СТРОКИ – НАЧАЛЬНЫЙ ИНДЕКС + 1, так как необходимо переслать байты, лежащие за начальным индексом. ЧИСЛО ВСТАВЛЯЕМЫХ БАЙТОВ – это, если не было усечения, длина подстроки. Если в результате вставки длина строки может быть больше максимальной, то это число равно максимальной длине строке минус ее текущая длина.

Примеры

1. ДЛИНА СТРОКИ = 20_{16} (32_{10}),
   НАЧАЛЬНЫЙ ИНДЕКС = 19_{16} (25_{10}),
   МАКСИМАЛЬНАЯ ДЛИНА = 30_{16} (48_{10}),
   ДЛИНА ПОДСТРОКИ = 06.

Таким образом, необходимо вставить подстроку длиной 6 байт, начиная с 25-го символа. Так как 8 байт необходимо переслать вверх (ЧИСЛО ПЕРЕСЫЛАЕМЫХ БАЙТОВ = 32 – 25 + 1), а 6 байт вставить, то время выполнения будет приблизительно

$39 * 8 + 39 * 6 + 300 = 312 + 234 + 300 = 846$ тактов (8080),
$40 * 8 + 40 * 6 + 288 = 320 + 240 + 288 = 848$ тактов (8085).

2. ДЛИНА СТРОКИ = 20_{16} (32_{10}),
   НАЧАЛЬНЫЙ ИНДЕКС = 19_{16} (25_{10}),
   МАКСИМАЛЬНАЯ ДЛИНА = 24_{16} (36_{10}),
   ДЛИНА ПОДСТРОКИ = 06.

В противоположность примеру 1 здесь можно вставить, не превышая максимальной длины строки, только 4 байта подстроки. Таким образом, ЧИСЛО ПЕРЕСЫЛАЕМЫХ БАЙТОВ = 8 и ЧИСЛО ВСТАВЛЯЕМЫХ БАЙТОВ = 4. Время выполнения будет приблизительно:

$$39 * 8 + 39 * 4 + 300 = 312 + 156 + 300 = 768 \text{ тактов } (8080),$$
$$40 * 8 + 40 * 4 + 288 = 320 + 160 + 288 = 768 \text{ тактов } (8085).$$

Размер программы: 103 байта.

Память, необходимая для данных: 1 байт в любом месте ОЗУ для флага ошибки (адрес INSERR).

Специальные случаи:

1. Если длина подстроки (вставка) равна 0, то осуществляется выход с очищенным флагом переноса (нет ошибок); строка при этом остается без изменения.

2. Если начальный индекс для вставки равен 0 (т. е. вставка должна начинаться с байта длины), то осуществляется выход с флагом переноса, установленным в 1 (указывает на ошибку); строка при этом остается без изменения.

3. Если строка со вставленной подстрокой превышает заданную максимальную длину, то вставляется только столько символов, сколько необходимо для получения максимальной длины. Флаг переноса устанавливается в 1 — это показывает, что вставка была усечена.

4. Если начальный индекс вставки выходит за конец строки, то вставка присоединяется к концу строки и флаг переноса устанавливается в 1, что указывает на ошибку.

5. Если начальная длина строки превышает ее заданную максимальную длину, то осуществляется выход с флагом переноса, установленным в 1 (указывает на ошибку); строка при этом остается без изменения.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес подстроки в регистрах D и E.
Базовый адрес строки в регистрах H и L.
Максимальная длина строки в регистре B.
Начальный индекс, с которого вставляется подстрока, в регистре C.

УСЛОВИЯ НА ВЫХОДЕ

Подстрока вставляется в строку. Если нет ошибок, флаг переноса очищается. Если начальный индекс равен 0 или длина подстроки равна 0, то флаг переноса устанавливается в 1, а строка не изменяется. Если начальный индекс больше длины строки, то флаг переноса устанавливается в 1, а подстрока присоединяется к концу строки. Если строка со вставленной подстрокой может превысить заданную максимальную длину, то флаг переноса устанавливается в 1 и в строку вставляются только те символы подстроки, которые доводят строку до максимальной длины.

ПРИМЕРЫ

1. Данные: строка = 0A'JOHN SMITH'($0A_{16} = 10_{10}$ — длина строки),
подстрока = 08'WILLIAM'(08 — длина подстроки),
максимальная длина строки = $14_{16} = 20_{10}$,
начальный индекс = 06.

Результат: строка = 12'JOHN WILLIAM SMITH'
($12_{16} = 18_{10}$ — длина строки с вставленной подстрокой),
флаг переноса = 0, так как при вставке не было проблем.

2. Данные: строка = 0A'JOHN SMITH' ($0A_{16} = 10_{10}$ – длина строки),
подстрока = 0C'ROCKEFELLER'
($0C_{16} = 12_{10}$ – длина подстроки),
максимальная длина строки = $14_{16} = 20_{10}$,
начальный индекс = 06.

Результат: строка = 14'JOHN ROCKEFELLESMITH'
($14_{16} = 20_{10}$ – длина строки, в которую вставлено столько символов подстроки, сколько позволила максимальная длина),
флаг переноса = 1, так как часть подстроки нельзя было вставить, не превысив максимальную длину строки.

```
;                                                                      ;
;                                                                      ;
;                                                                      ;
;                                                                      ;
;       ЗАГОЛОВОК:          ВСТАВКА ПОДСТРОКИ В СТРОКУ                 ;
;       ИМЯ:                INSERT                                     ;
;                                                                      ;
;                                                                      ;


;                                                                      ;
;       НАЗНАЧЕНИЕ:         ВСТАВЛЯЕТ ПОДСТРОКУ, ЗАДАННУЮ НАЧАЛЬНЫМ    ;
;                           ИНДЕКСОМ, В СТРОКУ                         ;
;                                                                      ;
;       ВХОД:               ПАРА РЕГИСТРОВ H = АДРЕС СТРОКИ            ;
;                           ПАРА РЕГИСТРОВ D = АДРЕС ПОДСТРОКИ, КОТОРАЯ;
;                                              ДОЛЖНА БЫТЬ  ВСТАВЛЕНА  ;
;                           РЕГИСТР B = МАКСИМАЛЬНАЯ ДЛИНА СТРОКИ      ;
;                           РЕГИСТР C = НАЧАЛЬНЫЙ ИНДЕКС В СТРОКЕ ДЛЯ  ;
;                                       ВСТАВКИ ПОДСТРОКИ              ;
;                                                                      ;
;                             МАКСИМАЛЬНАЯ ДЛИНА СТРОКИ 255 БАЙТ ПЛЮС ОДИН ;
;                             ПРЕДШЕСТВУЮЩИЙ БАЙТ С ДЛИНОЙ СТРОКИ      ;
;                                                                      ;
;       ВЫХОД:              ПОДСТРОКА ВСТАВЛЯЕТСЯ В СТРОКУ             ;
;                           ЕСЛИ НЕТ ОШИБОК, ТО                        ;
;                             ФЛАГ ПЕРЕНОСА := 0                       ;
;                           ИНАЧЕ                                      ;
;                             СЛЕДУЮЩИЕ УСЛОВИЯ ВЫЗЫВАЮТ ОШИБКУ И УСТАНОВКУ ;
;                             ФЛАГА ПЕРЕНОСА В 1:                      ;
;                             ИНДЕКС = 0; ПОДСТРОКА НЕ ВСТАВЛЯЕТСЯ     ;
;                             ДЛИНА СТРОКИ > МАКСИМАЛЬНОЙ ДЛИНЫ; ПОДСТРОКА ;
;                               НЕ ВСТАВЛЯЕТСЯ                         ;
;                             ИНДЕКС > ДЛИНЫ СТРОКИ; ПОДСТРОКА ДОБАВЛЯЕТСЯ ;
;                               В КОНЕЦ ИСХОДНОЙ СТРОКИ                ;
;                             ДЛИНА СТРОКИ + ДЛИНА ПОДСТРОКИ > МАКСИМАЛЬНОЙ ;
;                               ДЛИНЫ; ВСТАВЛЯЕТСЯ ТОЛЬКО ЧАСТЬ ПОДСТРОКИ ;
;                               ДО ДОСТИЖЕНИЯ МАКСИМАЛЬНОЙ ДЛИНЫ СТРОКИ ;
;                                                                      ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ                                     ;
;                                                                      ;
;       ВРЕМЯ:              ПРИБЛИЗИТЕЛЬНО                             ;
;                            39 * (ДЛИНА СТРОКИ - ИНДЕКС + 1) +        ;
;                            39 * ДЛИНА ПОДСТРОКИ ПЛЮС 300 ТАКТОВ ДЛЯ 8080 ;
;                           ПРИБЛИЗИТЕЛЬНО                             ;
;                            40 * (ДЛИНА СТРОКИ - ИНДЕКС + 1) +        ;
;                            40 * ДЛИНА ПОДСТРОКИ ПЛЮС 288 ТАКТОВ ДЛЯ 8085 ;
```

```
;                                                                  ;
;       РАЗМЕР:           ПРОГРАММА - 103 БАЙТА                    ;
;                         ДАННЫЕ    -   1 БАЙТ                     ;
;                                                                  ;
;                                                                  ;

INSERT:
        ;ЗАДАТЬ НАЧАЛЬНОЕ ЗНАЧЕНИЕ ФЛАГА ОШИБКИ
        SUB     A               ;ФЛАГ ОШИБКИ = 0 (НЕТ ОШИБОК)
        STA     INSERR

        ;ВЗЯТЬ ДЛИНЫ ПОДСТРОКИ И СТРОКИ
        ; ЕСЛИ ДЛИНА ПОДСТРОКИ = 0, ТО ВЫЙТИ, НЕ УКАЗЫВАЯ НА ОШИБКУ
        LDAX    D               ;ПРОВЕРИТЬ ДЛИНУ ПОДСТРОКИ
        ORA     A
        RZ                      ;ВЫЙТИ, ЕСЛИ ПОДСТРОКА ПУСТАЯ
                                ; ФЛАГ ПЕРЕНОСА = 0 (ОШИБОК НЕТ)

        ;ЕСЛИ НАЧАЛЬНЫЙ ИНДЕКС РАВЕН НУЛЮ, ТО ВЫЙТИ ПО ОШИБКЕ
IDX0:
        MOV     A,C             ;ПРОВЕРИТЬ НАЧАЛЬНЫЙ ИНДЕКС
        ORA     A
        STC                     ;СЧИТАТЬ, ЧТО ЕСТЬ ОШИБКА
        RZ                      ;ЕСЛИ ИНДЕКС = 0, ТО ВЫЙТИ С ОШИБКОЙ

        ;ПРОВЕРИТЬ, НЕ БУДЕТ ЛИ ПОСЛЕ ВСТАВКИ СТРОКА СЛИШКОМ ДЛИННОЙ.
        ; ЕСЛИ БУДЕТ, ОБРЕЗАТЬ СТРОКУ И УСТАНОВИТЬ ФЛАГ УСЕЧЕНИЯ.
        ;СТРОКА БУДЕТ СЛИШКОМ ДЛИННОЙ, ЕСЛИ ДЛИНА СТРОКИ + ДЛИНА
        ; ПОДСТРОКИ ПРЕВЫШАЮТ МАКСИМАЛЬНУЮ ДЛИНУ. НАПОМНИМ, ЧТО ДЛИНА
        ; СТРОКИ НЕ МОЖЕТ БЫТЬ БОЛЬШЕ 255 БАЙТ
CHKLEN:
        LDAX    D               ;ОБЩАЯ ДЛИНА = СТРОКА + ПОДСТРОКА
        ADD     M
        JC      TRUNC           ;ОБРЕЗАТЬ ПОДСТРОКУ, ЕСЛИ НОВАЯ ДЛИНА >
                                ; 255
        CMP     B               ;СРАВНИТЬ С МАКСИМАЛЬНОЙ ДЛИНОЙ СТРОКИ
        LDAX    D               ;A = ДЛИНА ПОДСТРОКИ
        JC      IDXLEN          ;ПЕРЕЙТИ, ЕСЛИ НОВАЯ ДЛИНА МЕНЬШЕ
        JZ      IDXLEN          ; МАКСИМАЛЬНОЙ ДЛИНЫ ИЛИ РАВНА ЕЙ

        ;ПОДСТРОКА НЕ УДОВЛЕТВОРЯЕТ, ПОЭТОМУ ОБРЕЗАТЬ ЕЕ.
        ; УСТАНОВИТЬ ФЛАГ ОШИБКИ, УКАЗЫВАЮЩИЙ, ЧТО СТРОКА ОБРЕЗАНА
        ;ПОДХОДЯЩАЯ ДЛИНА = МАКСИМАЛЬНАЯ ДЛИНА - ДЛИНА СТРОКИ
TRUNC:
        MVI     A,0FFH          ;УСТАНОВИТЬ ФЛАГ УСЕЧЕНИЯ ПОДСТРОКИ
        STA     INSERR
        MOV     A,B             ;ДЛИНА = МАКСИМАЛЬНАЯ ДЛИНА -
        SUB     M               ; ДЛИНА СТРОКИ
        RC                      ;ВЕРНУТЬСЯ ПО ОШИБКЕ, ЕСЛИ СТРОКА СЛИШКОМ
        STC                     ; ВЕЛИКА С САМОГО НАЧАЛА ИЛИ ЖЕ
        RZ                      ; МАКСИМАЛЬНАЯ ДЛИНА ТАКОВА, ЧТО НЕТ
                                ; МЕСТА ДЛЯ ПОДСТРОКИ

        ;ПРОВЕРИТЬ, НАХОДИТСЯ ЛИ ИНДЕКС ВНУТРИ СТРОКИ. ЕСЛИ НЕТ, ТО
        ; ПРИСОЕДИНИТЬ ПОДСТРОКУ В КОНЕЦ СТРОКИ
IDXLEN:
        MOV     B,A             ;B = ДЛИНА ПОДСТРОКИ
```

```
        MOV     A,M                 ;ВЗЯТЬ ДЛИНУ СТРОКИ
        CMP     C                   ;СРАВНИТЬ С ИНДЕКСОМ
        JNC     LENOK               ;ПЕРЕЙТИ, ЕСЛИ НАЧАЛЬНЫЙ ИНДЕКС
                                    ; НАХОДИТСЯ ВНУТРИ СТРОКИ

        ;ИНДЕКС НЕ ВНУТРИ СТРОКИ, ТОГДА ПРИСОЕДИНИТЬ ПОДСТРОКУ
        ; НОВАЯ ДЛИНА СТРОКИ = СТАРАЯ ДЛИНА + ДЛИНА ПОДСТРОКИ
        MOV     C,A                 ;СОХРАНИТЬ ТЕКУЩУЮ ДЛИНУ СТРОКИ
        ADD     B                   ;ПРИБАВИТЬ ДЛИНУ ПОДСТРОКИ
        MOV     M,A                 ;УСТАНОВИТЬ НОВУЮ ДЛИНУ СТРОКИ

        ;УСТАНОВИТЬ АДРЕСА ДЛЯ ОБ'ЕДИНЕНИЯ
        ; DE = АДРЕС СТРОКИ + ДЛИНА СТРОКИ + 1
        ; HL = АДРЕС ПОДСТРОКИ
        XCHG                        ;HL = АДРЕС ПОДСТРОКИ
        MOV     A,C                 ;DE = КОНЕЦ СТРОКИ
        INR     A
        ADD     E
        MOV     E,A
        JNC     IDXL1
        INR     D
IDXL1:
        MVI     A,0FFH              ;УСТАНОВИТЬ ФЛАГ ОШИБКИ
        STA     INSERR
        JMP     MVESUB              ;ПРОСТО ПЕРЕСЫЛАТЬ, НЕ РАЗДВИГАЯ СТРОКУ

        ;ОСВОБОДИТЬ МЕСТО В ИСХОДНОЙ СТРОКЕ ДЛЯ ПОДСТРОКИ, ДЛЯ ЧЕГО
        ; ПЕРЕСЛАТЬ СИМВОЛЫ, НАЧИНАЯ С КОНЦА СТРОКИ ДО ИНДЕКСА, ВВЕРХ
        ; НА РАЗМЕР ПОДСТРОКИ
        ; A = ДЛИНА СТРОКИ
LENOK:
        PUSH    B                   ;СОХРАНИТЬ ДЛИНУ ПОДСТРОКИ
        PUSH    D                   ;СОХРАНИТЬ АДРЕС ПОДСТРОКИ

        ;НОВАЯ ДЛИНА СТРОКИ = СТАРАЯ ДЛИНА + ДЛИНА ПОДСТРОКИ
        MOV     E,A                 ;DE = ДЛИНА СТРОКИ
        MVI     D,0
        ADD     B
        MOV     M,A                 ;ЗАПОМНИТЬ НОВУЮ ДЛИНУ СТРОКИ

        ;ВЫЧИСЛИТЬ ЧИСЛО ПЕРЕСЫЛАЕМЫХ СИМВОЛОВ
        ; = ДЛИНА СТРОКИ - НАЧАЛЬНЫЙ ИНДЕКС + 1
        MOV     A,E                 ;ВЗЯТЬ ПЕРВОНАЧАЛЬНУЮ ДЛИНУ СТРОКИ
        SUB     C
        INR     A                   ;A = ЧИСЛО ПЕРЕСЫЛАЕМЫХ СИМВОЛОВ

        ;ВЫЧИСЛИТЬ АДРЕС ПОСЛЕДНЕГО СИМВОЛА В СТРОКЕ. ОН ЯВЛЯЕТСЯ
        ; ИСХОДНЫМ АДРЕСОМ ДЛЯ ПЕРЕСЫЛКИ И РАВЕН СУММЕ АДРЕСА СТРОКИ
        ; И ДЛИНЫ СТРОКИ
        DAD     D                   ;HL, КАК И DE, УКАЗЫВАЕТ НА
        MOV     E,L                 ; ПОСЛЕДНИЙ СИМВОЛ В СТРОКЕ
        MOV     D,H

        ;ВЫЧИСЛИТЬ АДРЕС НАЗНАЧЕНИЯ
        ; = АДРЕС СТРОКИ + ДЛИНА СТРОКИ + ДЛИНА ПОДСТРОКИ
        MOV     C,B                 ;BC = ДЛИНА ПОДСТРОКИ
        MVI     B,0
        DAD     B                   ;HL = АДРЕС НАЗНАЧЕНИЯ
```

```
                                        ;DE = ИСХОДНЫЙ АДРЕС
        MOV     C,A                     ;C = ЧИСЛО ПЕРЕСЫЛАЕМЫХ СИМВОЛОВ
OPNLP:
        LDAX    D                       ;ВЗЯТЬ СИМВОЛ
        MOV     M,A                     ;ПЕРЕСЛАТЬ ЕГО ВВЕРХ
        DCX     H                       ;ИЗМЕНИТЬ АДРЕСА ДЛЯ СЛЕДУЮЩЕГО
        DCX     D                       ; СИМВОЛА
        DCR     C
        JNZ     OPNLP                   ;ПРОДОЛЖАТЬ, ПОКА НЕ БУДУТ ПЕРЕСЛАНЫ
                                        ; ВСЕ СИМВОЛЫ

        ;ВОССТАНОВИТЬ РЕГИСТРЫ
        INX     D                       ;DE = АДРЕС, ПО КОТОРОМУ ПЕРЕСЫЛАЕТСЯ
                                        ; СТРОКА
        POP     H                       ;HL = АДРЕС ПОДСТРОКИ
        POP     B                       ;B = ДЛИНА ПОДСТРОКИ

        ;ПЕРЕСЛАТЬ ПОДСТРОКУ В ОСВОБОЖДЕННУЮ ОБЛАСТЬ
        ; HL = АДРЕС ПОДСТРОКИ
        ; DE = АДРЕС, ПО КОТОРОМУ ПЕРЕСЫЛАЕТСЯ ПОДСТРОКА
        ; B = ДЛИНА ПОДСТРОКИ
MVESUB:
        INX     H                       ;УВЕЛИЧИТЬ АДРЕС, ЧТОБЫ ПРОПУСТИТЬ
                                        ; БАЙТ ДЛИНЫ ПОДСТРОКИ
MVELP:
        MOV     A,M                     ;ВЗЯТЬ СИМВОЛ ИЗ ПОДСТРОКИ
        STAX    D                       ;ВСТАВИТЬ ЕГО В ОСВОБОЖДЕННУЮ ОБЛАСТЬ
        INX     H
        INX     D
        DCR     B
        JNZ     MVELP
        LDA     INSERR                  ;ВЗЯТЬ ФЛАГ ОШИБКИ
        RAR                             ;ЕСЛИ INSERR <> 0, ТО ФЛАГ ПЕРЕНОСА = 1,
                                        ; УКАЗЫВАЯ ТЕМ САМЫМ НА НАЛИЧИЕ ОШИБКИ
        RET

        ;СЕКЦИЯ ДАННЫХ
INSERR: DS      1                       ;ФЛАГ, ИСПОЛЬЗУЕМЫЙ ДЛЯ ИНДИКАЦИИ
                                        ; ОШИБКИ


;                                                                       ;
;                                                                       ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                               ;
;                                                                       ;
;                                                                       ;

SC8F:
        LXI     H,STG   ;HL = БАЗОВЫЙ АДРЕС СТРОКИ
        LXI     D,SSTG  ;DE = БАЗОВЫЙ АДРЕС ПОДСТРОКИ
        LDA     IDX
        MOV     C,A     ;C = НАЧАЛЬНЫЙ ИНДЕКС ДЛЯ ВСТАВКИ
        LDA     MXLEN
        MOV     B,A     ;B = МАКСИМАЛЬНАЯ ДЛИНА СТРОКИ
        CALL    INSERT  ;ВСТАВИТЬ ПОДСТРОКУ
                        ;РЕЗУЛЬТАТ ВСТАВКИ '-' В '123456' ПО
                        ; ИНДЕКСУ 1 - "-123456"
```

```
        JMP     SC8F    ;ЦИКЛ ДЛЯ СЛЕДУЮЩЕГО ТЕСТА

        ;СЕКЦИЯ ДАННЫХ
IDX:    DB      1                   ;НАЧАЛЬНЫЙ ИНДЕКС ДЛЯ ВСТАВКИ
MXLEN:  DB      20H                 ;МАКСИМАЛЬНАЯ ДЛИНА РЕЗУЛЬТАТА
STG:    DB      06H                 ;ДЛИНА СТРОКИ
        DB      '123456                          ' ;МАКС.ДЛИНА 32 БАЙТА
SSTG:   DB      1                   ;ДЛИНА ПОДСТРОКИ
        DB      '-                               ' ;МАКС.ДЛИНА 32 БАЙТА

        END
```

ГЛАВА 9

ОПЕРАЦИИ С МАССИВАМИ

9А. СУММИРОВАНИЕ 8-РАЗРЯДНОГО МАССИВА (ASUM8)

Суммируются элементы массива с получением 16-разрядного результата. Массив содержит до 255 элементов длиной в байт.

Процедура. Сначала очищается сумма. Затем к младшему по значению байту суммы прибавляется по одному элементу за раз, начиная с базового адреса. Как только сложение вызывает перенос, увеличивается старший по значению байт суммы.

Используемые регистры: AF, B, DE, HL.
Время выполнения: приблизительно 37 тактов на элемент длиной в байт плюс 57 тактов (8080) или 51 такт (8085).
Размер программы: 22 байта.
Память, необходимая для данных: отсутствует.
Специальный случай: размер массива, равный 0, вызывает немедленный выход с суммой, равной 0.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес массива в регистрах H и L.
Размер массива в байтах в регистре B.

УСЛОВИЯ НА ВЫХОДЕ

Сумма в регистрах H и L.

ПРИМЕР

1. Данные: массив содержит $E7_{16}$ $5A_{16}$ 23_{16} 16_{16} 31_{16} CB_{16} 70_{16} $E1_{16}$.
   Результат: сумма = (H и L) = $03D7_{16}$.

```
;                                                               ;
;                                                               ;
;                                                               ;
;                                                               ;
;       ЗАГОЛОВОК:      СУММИРОВАНИЕ 8-РАЗРЯДНОГО МАССИВА       ;
;       ИМЯ:            ASUM8                                   ;
```

```
;                                                                    ;
;                                                                    ;

;                                                                    ;
;       НАЗНАЧЕНИЕ:      СУММИРУЕТ ЭЛЕМЕНТЫ МАССИВА С ПОЛУЧЕНИЕМ     ;
;                        16-РАЗРЯДНОГО РЕЗУЛЬТАТА. МАКСИМАЛЬНЫЙ РАЗМЁР - ;
;                        255 БАЙТ                                    ;
;                                                                    ;
;       ВХОД:            ПАРА РЕГИСТРОВ H = БАЗОВЫЙ АДРЕС МАССИВА    ;
;                        РЕГИСТР B = РАЗМЕР МАССИВА В БАЙТАХ         ;
;                                                                    ;
;       ВЫХОД:           ПАРА РЕГИСТРОВ H = СУММА                    ;
;                                                                    ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF,B,DE,HL                            ;
;                                                                    ;
;       ВРЕМЯ:           ПРИБЛИЗИТЕЛЬНО 37 ТАКТОВ НА ЭЛЕМЕНТ МАССИВА ;
;                        ПЛЮС 57 ТАКТОВ ДЛЯ 8080                     ;
;                        ПРИБЛИЗИТЕЛЬНО 37 ТАКТОВ НА ЭЛЕМЕНТ МАССИВА ;
;                        ПЛЮС 51 ТАКТ ДЛЯ 8085                       ;
;                                                                    ;
;       РАЗМЕР:          ПРОГРАММА - 22 БАЙТА                        ;
;                                                                    ;
;                                                                    ;

ASUM8:
        ;ПРОВЕРИТЬ ДЛИНУ МАССИВА
        ;ЕСЛИ В МАССИВЕ НИЧЕГО НЕТ, ТО ВЫЙТИ С СУММОЙ = 0
        XCHG                    ;СОХРАНИТЬ БАЗОВЫЙ АДРЕС МАССИВА
        LXI     H,0             ;ЗАДАТЬ НАЧАЛЬНОЕ ЗНАЧЕНИЕ СУММЫ = 0

        ;ПРОВЕРИТЬ ДЛИНУ МАССИВА НА РАВЕНСТВО НУЛЮ
        MOV     A,B             ;ПРОВЕРИТЬ ДЛИНУ МАССИВА
        ORA     A
        RZ                      ;ЕСЛИ ДЛИНА = 0, ТО ВЫЙТИ С СУММОЙ = 0

        ;ЗАДАТЬ НАЧАЛЬНЫЕ ЗНАЧЕНИЯ УКАЗАТЕЛЯ МАССИВА И СУММЫ
        XCHG                    ;ВОССТАНОВИТЬ БАЗОВЫЙ АДРЕС В H И L
                                ; СТАРШИЙ БАЙТ СУММЫ = 0
        SUB     A               ;A = МЛАДШИЙ БАЙТ СУММЫ = 0
                                ;D = СТАРШИЙ БАЙТ СУММЫ

        ;ПРИБАВЛЯТЬ К СУММЕ ЭЛЕМЕНТЫ ДЛИНОЙ В БАЙТ, ПО ОДНОМУ ЗА РАЗ
        ;УВЕЛИЧИВАТЬ СТАРШИЙ БАЙТ СУММЫ НА 1 КАЖДЫЙ РАЗ ПРИ ПЕРЕНОСЕ
SUMLP:
        ADD     M               ;ПРИБАВИТЬ СЛЕДУЮЩИЙ ЭЛЕМЕНТ К МЛАДШЕМУ
                                ; БАЙТУ СУММЫ
        JNC     DECCNT          ;ПЕРЕЙТИ, ЕСЛИ НЕТ ПЕРЕНОСА
        INR     D               ; ИНАЧЕ УВЕЛИЧИТЬ СТАРШИЙ БАЙТ СУММЫ НА 1
DECCNT:
        INX     H
        DCR     B
        JNZ     SUMLP
EXIT:
        MOV     L,A             ;HL = СУММА
        MOV     H,D
        RET
```

```
;                                                                    ;
;                                                                    ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                            ;
;                                                                    ;
;                                                                    ;

SC9A:
        LXI     H,BUF           ;HL = БАЗОВЫЙ АДРЕС БУФЕРА
        LDA     BUFSZ
        MOV     B,A             ;B = РАЗМЕР БУФЕРА В БАЙТАХ
        CALL    ASUM8
                                ;СУММА ТЕСТОВЫХ ДАННЫХ РАВНА
                                ; ШЕСТНАДЦАТЕРИЧНОМУ ЧИСЛУ 07F8,
                                ; РЕГИСТРЫ H И L = 07F8H

        JMP     SC9A

        ;ДАННЫЕ ДЛЯ ТЕСТА, КОТОРЫЕ МОЖНО ИЗМЕНИТЬ НА ДРУГИЕ ЗНАЧЕНИЯ
SIZE    EQU     010H            ;РАЗМЕР БУФЕРА В БАЙТАХ
BUFSZ:  DB      SIZE            ;РАЗМЕР БУФЕРА В БАЙТАХ

BUF:    DB      00H             ;БУФЕР
        DB      11H             ;ДЕСЯТИЧНЫЕ ЗНАЧЕНИЯ ЭЛЕМЕНТОВ:
        DB      22H             ; 0,17,34,51,68,85,102,119,136,153,170
        DB      33H             ; 187,204,221,238,255
        DB      44H
        DB      55H
        DB      66H
        DB      77H
        DB      88H
        DB      99H
        DB      0AAH
        DB      0BBH
        DB      0CCH
        DB      0DDH
        DB      0EEH
        DB      0FFH            ;СУММА = 07F8 (ДЕСЯТИЧНОЕ ЧИСЛО 2040)

        END
```

9B. СУММИРОВАНИЕ 16-РАЗРЯДНОГО МАССИВА (ASUM16)

Суммируются элементы массива с получением 24-разрядного результата. Массив содержит до 255 элементов длиной в слово (16-разрядов). Элементы расположены в обычном для 8080, 8085 формате, при котором младшие по значению байты идут первыми.

Процедура. Сначала очищается сумма. Затем к младшему по значению байту суммы прибавляется по одному элементу за один раз, начиная с базового адреса. Как только сложение вызывает перенос, увеличивается старший по значению байт суммы.

Используемые регистры: все.
Время выполнения: приблизительно 69 (8080) или 66 (8085) тактов на 16-разрядный элемент плюс 57 (8080) или 51 (8085) тактов.
Размер программы: 28 байт.

Память, необходимая для данных: отсутствует.
Специальный случай: размер массива, равный 0, вызывает немедленный выход с суммой, равной 0.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес массива в регистрах H и L.
Размер массива в 16-разрядных словах в регистре B.

УСЛОВИЯ НА ВЫХОДЕ

Старший по значению байт суммы в регистре E.
Средний и младший по значению байты суммы в регистрах H и L.

ПРИМЕР

1. Данные: массив (16-разрядных слов) содержит $F7A1_{16}$ $239B_{16}$ $31D5_{16}$ $70F2_{16}$ $5A36_{16}$ $166C_{16}$ $CBF5_{16}$ $E107_{16}$.
   Результат: сумма = $03DBA1_{16}$,
   (E) = 03_{16},
   (H и L) = $DBA1_{16}$.

```
;                                                              ;
;                                                              ;
;                                                              ;
;                                                              ;
;       ЗАГОЛОВОК:      СУММИРОВАНИЕ 16-РАЗРЯДНОГО МАССИВА     ;
;       ИМЯ:            ASUM16                                 ;
;                                                              ;
;                                                              ;


;                                                              ;
;       НАЗНАЧЕНИЕ:     СУММИРУЕТ ЭЛЕМЕНТЫ МАССИВА С ПОЛУЧЕНИЕМ ;
;                       24-РАЗРЯДНОГО РЕЗУЛЬТАТА. МАКСИМАЛЬНЫЙ РАЗМЕР - ;
;                       255 16-РАЗРЯДНЫХ ЭЛЕМЕНТОВ             ;
;                                                              ;
;       ВХОД:           ПАРА РЕГИСТРОВ H = БАЗОВЫЙ АДРЕС МАССИВА ;
;                       РЕГИСТР B = РАЗМЕР МАССИВА В СЛОВАХ    ;
;                                                              ;
;       ВЫХОД:          РЕГИСТР A = СТАРШИЙ БАЙТ СУММЫ         ;
;                       РЕГИСТР H = СРЕДНИЙ БАЙТ СУММЫ         ;
;                       РЕГИСТР L = МЛАДШИЙ БАЙТ СУММЫ         ;
;                                                              ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ                             ;
;                                                              ;
;       ВРЕМЯ:          ПРИБЛИЗИТЕЛЬНО 69 ТАКТОВ НА ЭЛЕМЕНТ МАССИВА ;
;                       ПЛЮС 57 ТАКТОВ ДЛЯ 8080                ;
;                       ПРИБЛИЗИТЕЛЬНО 66 ТАКТОВ НА ЭЛЕМЕНТ МАССИВА ;
;                       ПЛЮС 51 ТАКТ ДЛЯ 8085                  ;
;                                                              ;
;       РАЗМЕР:         ПРОГРАММА - 28 БАЙТА                   ;
;                                                              ;
;                                                              ;

ASUM16:
        ;ПРОВЕРИТЬ ДЛИНУ МАССИВА
```

```
        ;ЕСЛИ В МАССИВЕ НИЧЕГО НЕТ, ТО ВЫЙТИ С СУММОЙ = 0
        XCHG                            ;СОХРАНИТЬ БАЗОВЫЙ АДРЕС МАССИВА
        LXI     H,0                     ;ЗАДАТЬ НАЧАЛЬНОЕ ЗНАЧЕНИЕ СУММЫ = 0

        ;ПРОВЕРИТЬ ДЛИНУ МАССИВА НА РАВЕНСТВО НУЛЮ
        MOV     A,B                     ;ПРОВЕРИТЬ ДЛИНУ МАССИВА
        ORA     A
        RZ                              ;ЕСЛИ ДЛИНА = 0, ТО ВЫЙТИ С СУММОЙ = 0

        ;ЗАДАТЬ НАЧАЛЬНЫЕ ЗНАЧЕНИЯ УКАЗАТЕЛЯ МАССИВА И СУММЫ
        XCHG                            ;ПЕРЕСЛАТЬ БАЗОВЫЙ АДРЕС В HL
                                        ; МЛАДШИЙ, СРЕДНИЙ БАЙТЫ СУММЫ = 0
        MOV     C,E                     ;СТАРШИЙ БАЙТ СУММЫ = 0
                                        ;C = СТАРШИЙ БАЙТ СУММЫ
                                        ;D = СРЕДНИЙ БАЙТ СУММЫ
                                        ;E = МЛАДШИЙ БАЙТ СУММЫ

        ;ПРИБАВЛЯТЬ К СУММЕ ЭЛЕМЕНТЫ ДЛИНОЙ В СЛОВО, ПО ОДНОМУ ЗА РАЗ
        ;УВЕЛИЧИВАТЬ СТАРШИЙ БАЙТ СУММЫ НА 1 КАЖДЫЙ РАЗ ПРИ ПЕРЕНОСЕ
SUMLP:
        MOV     A,E                     ;СЛОЖИТЬ МЛАДШИЕ БАЙТЫ ЭЛЕМЕНТА И СУММЫ
        ADD     M
        MOV     E,A
        INX     H
        MOV     A,D                     ;ПРИБАВИТЬ СТАРШИЙ БАЙТ ЭЛЕМЕНТА
        ADC     M                       ; К СРЕДНЕМУ БАЙТУ СУММЫ
        MOV     D,A
        JNC     DECCNT                  ;ПЕРЕЙТИ, ЕСЛИ НЕТ ПЕРЕНОСА
        INR     C                       ; ИНАЧЕ УВЕЛИЧИТЬ НА 1 СТАРШИЙ БАЙТ СУММЫ
DECCNT:
        INX     H
        DCR     B
        JNZ     SUMLP
EXIT:
        XCHG                            ;HL = СРЕДНИЙ И МЛАДШИЙ БАЙТЫ СУММЫ
        MOV     A,C                     ;A = СТАРШИЙ БАЙТ СУММЫ
        RET



;                                                                       ;
;                                                                       ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                               ;
;                                                                       ;
;                                                                       ;

SC9B:
        LXI     H,BUF                   ;HL = БАЗОВЫЙ АДРЕС МАССИВА
        LDA     BUFSZ
        MOV     B,A                     ;B = РАЗМЕР МАССИВА В СЛОВАХ
        CALL    ASUM16
                                        ;СУММА ТЕСТОВЫХ ДАННЫХ РАВНА
                                        ; ШЕСТНАДЦАТЕРИЧНОМУ ЧИСЛУ 31FF8,
                                        ; РЕГИСТРЫ H И L = 1FF8H
                                        ; РЕГИСТР A = 3

        JMP     SC9B
```

```
        ;ДАННЫЕ ДЛЯ ТЕСТА, КОТОРЫЕ МОЖНО ЗАМЕНИТЬ ДРУГИМИ ЗНАЧЕНИЯМИ
SIZE    EQU     010H                ;РАЗМЕР МАССИВА В СЛОВАХ
BUFSZ:  DB      SIZE                ;РАЗМЕР МАССИВА В СЛОВАХ

BUF:    DW      000H                ;БУФЕР
        DW      111H                ;ДЕСЯТИЧНЫЕ ЗНАЧЕНИЯ ЭЛЕМЕНТОВ:
        DW      222H                ; 0,273,546,819,1092,1365,1638,1911
        DW      333H                ; 2184,2457,2730,3003,3276,56797
        DW      444H                ; 61166,65535
        DW      555H
        DW      666H
        DW      777H
        DW      888H
        DW      999H
        DW      0AAAH
        DW      0BBBH
        DW      0CCCH
        DW      0DDDDH
        DW      0EEEEH
        DW      0FFFFH              ;СУММА = 31FFB (ДЕСЯТИЧНОЕ ЧИСЛО 204792)

        END
```

9С. ПОИСК МАКСИМАЛЬНОГО ЭЛЕМЕНТА ДЛИНОЙ 1 БАЙТ (MAXELM)

Ищется максимальный элемент в массиве. Массив содержит до 255 элементов длиной в байт.

Процедура. Если в массиве нет элементов, то осуществляется немедленный выход из программы (флаг переноса устанавливается в 1). В противном случае считается, что элемент, находящийся по базовому адресу, является максимальным. Затем просматривается весь массив, при этом сравнивается предположительный максимум с каждым элементом и сохраняются наибольшее значение и его адрес. В конце очищается флаг переноса для указания правильности результата.

Используемые регистры: AF, B, DE, HL.
Время выполнения: приблизительно от 37 до 62 тактов на элемент плюс дополнительно 36 тактов (8080) или от 34 до 53 тактов на элемент плюс дополнительно 36 тактов (8085). Если должен заменяться максимум в среднем в половине итераций, то время выполнения приблизительно равно 99 * РАЗМЕР МАССИВА/2 + + 36 тактов (8080) или 87 * РАЗМЕР МАССИВА/2 + 36 тактов (8085).
Размер программы: 22 байта.
Память, необходимая для данных: отсутствует.
Специальные случаи:

1. Размер массива, равный 0, вызывает немедленный выход с флагом переноса, установленным в 1, что указывает на неправильный результат.

2. Если оказывается, что есть больше одного наибольшего беззнакового значения, то возвращается наименьший возможный адрес. Таким образом, возвращается адрес максимального значения, которое расположено ближе к началу массива.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес массива в регистрах H и L.
Размер массива в байтах в регистре B.

УСЛОВИЯ НА ВЫХОДЕ

Наибольший беззнаковый элемент в регистре A.

Адрес наибольшего беззнакового элемента в регистрах H и L.

Флаг переноса = 0, если результат правильный; 1, если размер массива равен 0, при этом результат не имеет смысла.

ПРИМЕР

1. Данные: массив (байтов) содержит 35_{16} $A6_{16}$ $D2_{16}$ $1B_{16}$ 44_{16} 59_{16} $7A_{16}$ CF_{16}.
   Результат: наибольшим беззнаковым элементом является элемент номер 2 ($D2_{16}$),
   (A) = наибольший элемент ($D2_{16}$),
   (H и L) = БАЗА + 2 (наименьший адрес, содержащий $D2_{16}$);
   флаг переноса = 0, что указывает на неравенство нулю размера массива и на правильность результата.

```
;                                                                       ;
;                                                                       ;
;                                                                       ;
;                                                                       ;
;       ЗАГОЛОВОК:      ПОИСК МАКСИМАЛЬНОГО ЭЛЕМЕНТА ДЛИНОЙ 1 БАЙТ      ;
;       ИМЯ:            MAXELM                                          ;
;                                                                       ;
;                                                                       ;

;                                                                       ;
;       НАЗНАЧЕНИЕ:     ПО ЗАДАННЫМ БАЗОВОМУ АДРЕСУ И РАЗМЕРУ МАССИВА   ;
;                       НАХОДИТ НАИБОЛЬШИЙ ЭЛЕМЕНТ                      ;
;                                                                       ;
;       ВХОД:           ПАРА РЕГИСТРОВ H = БАЗОВЫЙ АДРЕС МАССИВА        ;
;                       РЕГИСТР B = РАЗМЕР МАССИВА В БАЙТАХ             ;
;                                                                       ;
;       ВЫХОД:          ЕСЛИ РАЗМЕР МАССИВА НЕ НУЛЬ, ТО                 ;
;                         ФЛАГ ПЕРЕНОСА = 0                             ;
;                         РЕГИСТР A = НАИБОЛЬШИЙ ЭЛЕМЕНТ                ;
;                         ПАРА РЕГИСТРОВ H = АДРЕС ЭТОГО ЭЛЕМЕНТА       ;
;                          ЕСЛИ ЕСТЬ ЕЩЕ ЭЛЕМЕНТЫ С ТАКИМ ЖЕ ЗНАЧЕНИЕМ, ;
;                          ТО ПАРА РЕГИСТРОВ H СОДЕРЖИТ АДРЕС, БЛИЖАЙШИЙ;
;                          К БАЗОВОМУ                                   ;
;                       ИНАЧЕ                                           ;
;                         ФЛАГ ПЕРЕНОСА = 1                             ;
;                                                                       ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF,B,DE,HL                               ;
;                                                                       ;
;       ВРЕМЯ:          ПРИБЛИЗИТЕЛЬНО ОТ 37 ДО 62 ТАКТОВ НА БАЙТ       ;
;                       ПЛЮС 36 ТАКТОВ ДЛЯ 8080                         ;
;                       ПРИБЛИЗИТЕЛЬНО ОТ 34 ДО 53 ТАКТОВ НА БАЙТ       ;
;                       ПЛЮС 36 ТАКТОВ ДЛЯ 8085                         ;
;                                                                       ;
;       РАЗМЕР:         ПРОГРАММА - 22 БАЙТА                            ;
;                                                                       ;
;                                                                       ;

MAXELM:
        ;ЕСЛИ НЕТ ЭЛЕМЕНТОВ В МАССИВЕ, ТО ВЫЙТИ С УСТАНОВЛЕННЫМ
        ; ФЛАГОМ ПЕРЕНОСА
        MOV     A,B                     ;ПРОВЕРИТЬ РАЗМЕР МАССИВА
```

```
        ORA     A
        STC                     ;УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА, ЧТОБЫ
                                ; УКАЗАТЬ НА ВЫХОД ПО ОШИБКЕ
        RZ                      ;ВЕРНУТЬСЯ, ЕСЛИ НЕТ ЭЛЕМЕНТОВ

        ;ЗАМЕНИТЬ ПРЕДЫДУЩИЙ ЭЛЕМЕНТ, СЧИТАВШИЙСЯ НАИБОЛЬШИМ,
        ; ТЕКУЩИМ. ПЕРВЫЙ РАЗ НАИБОЛЬШИМ СЧИТАЕТСЯ ПЕРВЫЙ ЭЛЕМЕНТ
MAXLP:  MOV     A,M             ;НАИБОЛЬШИЙ = ТЕКУЩИЙ ЭЛЕМЕНТ
        MOV     E,L             ;СОХРАНИТЬ ЕГО АДРЕС
        MOV     D,H

        ;СРАВНИТЬ ТЕКУЩИЙ ЭЛЕМЕНТ С НАИБОЛЬШИМ
        ;ПРОДОЛЖАТЬ СРАВНЕНИЕ, ПОКА ТЕКУЩИЙ ЭЛЕМЕНТ БОЛЬШЕ
MAXLP1:
        DCR     B
        JZ      EXIT
        INX     H
        CMP     M               ;СРАВНИТЬ ТЕКУЩИЙ ЭЛЕМЕНТ С НАИБОЛЬШИМ
        JNC     MAXLP1          ;ПРОДОЛЖАТЬ, ПОКА ТЕКУЩИЙ ЭЛЕМЕНТ БОЛЬШЕ
        JMP     MAXLP           ;ИНАЧЕ ЗАМЕНИТЬ НАИБОЛЬШИЙ ЭЛЕМЕНТ
EXIT:
        ORA     A               ;ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА - НЕТ ОШИБКИ
        XCHG                    ;HL = АДРЕС НАИБОЛЬШЕГО ЭЛЕМЕНТА
        RET
;                                                                       ;
;                                                                       ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                               ;
;                                                                       ;
;                                                                       ;
SC9C:
        LXI     H,ARY           ;HL = БАЗОВЫЙ АДРЕС МАССИВА
        MVI     B,SZARY         ;B = РАЗМЕР МАССИВА В БАЙТАХ
        CALL    MAXELM
                                ;РЕЗУЛЬТАТ ДЛЯ ТЕСТОВЫХ ДАННЫХ:
                                ; A = ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО FF
                                ; (МАКСИМАЛЬНЫЙ ЭЛЕМЕНТ),
                                ; HL = АДРЕС FF В МАССИВЕ

        JMP     SC9C            ;ЦИКЛ ДЛЯ СЛЕДУЮЩИХ ТЕСТОВ

SZARY   EQU     10H             ;РАЗМЕР МАССИВА В БАЙТАХ
ARY:    DB      8
        DB      7
        DB      6
        DB      5
        DB      4
        DB      3
        DB      2
        DB      1
        DB      0FFH
        DB      0FEH
        DB      0FDH
        DB      0FCH
        DB      0FBH
        DB      0FAH
        DB      0F9H
        DB      0F8H

        END
```

9D. ПОИСК МИНИМАЛЬНОГО ЭЛЕМЕНТА ДЛИНОЙ 1 БАЙТ (MINELM)

Ищется минимальный элемент в массиве. Массив содержит до 255 элементов длиной в байт.

Процедура. Если в массиве нет элементов, то осуществляется немедленный выход из программы (флаг переноса устанавливается в 1). В противном случае считается, что элемент, находящийся по базовому адресу, является минимальным. Затем просматривается весь массив, при этом предположительный минимум сравнивается с каждым элементом и сохраняются наименьшее значение и его адрес. В конце очищается флаг переноса для указания правильности результата.

Используемые регистры: AF, B, DE, HL.
Время выполнения: приблизительно от 37 до 72 тактов на элемент плюс дополнительно 36 тактов (8080) или от 34 до 63 тактов на элемент плюс дополнительно 36 тактов (8085). **Если минимум должен заменяться в среднем в половине итераций**, то время выполнения приблизительно равно 109 * РАЗМЕР МАССИВА/2 + + 36 тактов (8080) или 97 * РАЗМЕР МАССИВА + 36 тактов (8085).
Размер программы: 25 байт.
Память, необходимая для данных: отсутствует.
Специальные случаи:
1. Размер массива, равный 0, вызывает немедленный выход с флагом переноса, установленным в 1, что указывает на неправильный результат.
2. Если оказывается, что есть больше одного наименьшего значения без знака, то возвращается наименьший возможный адрес. Таким образом, возвращается адрес минимального значения, которое расположено ближе всего к началу массива.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес массива в регистрах H и L.
Размер массива в байтах в регистре B.

УСЛОВИЯ НА ВЫХОДЕ

Наименьший беззнаковый элемент в регистре A.
Адрес наименьшего беззнакового элемента в регистрах H и L.
Флаг переноса = 0, если результат правильный; 1, если размер массива = 0, при этом результат не имеет смысла.

ПРИМЕР

1. Данные: массив (байтов) содержит 35_{16} $A6_{16}$ $D2_{16}$ $1B_{16}$ 44_{16} 59_{16} $7A_{16}$ CF_{16}.
Результат: наименьшим беззнаковым элементом является элемент номер 3 ($1B_{16}$),
(A) – наименьший элемент ($1B_{16}$),
(H и L) = БАЗА + 3 (наименьший адрес, содержащий $1B_{16}$),
флаг переноса = 0, что указывает на неравенство нулю размера массива и правильность результата.

```
;                                                                   ;
;                                                                   ;
;                                                                   ;
;                                                                   ;
;       ЗАГОЛОВОК:      ПОИСК МИНИМАЛЬНОГО ЭЛЕМЕНТА ДЛИНОЙ 1 БАЙТ   ;
;       ИМЯ:            MINELM                                      ;
```

```
;                                                                          ;
;                                                                          ;

;                                                                          ;
;       НАЗНАЧЕНИЕ:     ПО ЗАДАННЫМ БАЗОВОМУ АДРЕСУ И РАЗМЕРУ МАССИВА      ;
;                       НАХОДИТ НАИМЕНЬШИЙ ЭЛЕМЕНТ                         ;
;                                                                          ;
;       ВХОД:           ПАРА РЕГИСТРОВ H = БАЗОВЫЙ АДРЕС МАССИВА           ;
;                       РЕГИСТР B = РАЗМЕР МАССИВА В БАЙТАХ                ;
;                                                                          ;
;       ВЫХОД:          ЕСЛИ РАЗМЕР МАССИВА НЕ НУЛЬ, ТО                    ;
;                         ФЛАГ ПЕРЕНОСА = 0                                ;
;                         РЕГИСТР A = НАИМЕНЬШИЙ ЭЛЕМЕНТ                   ;
;                         ПАРА РЕГИСТРОВ H = АДРЕС ЭТОГО ЭЛЕМЕНТА          ;
;                          ЕСЛИ ЕСТЬ ЕЩЕ ЭЛЕМЕНТЫ С ТАКИМ ЖЕ ЗНАЧЕНИЕМ,    ;
;                          ТО ПАРА РЕГИСТРОВ H СОДЕРЖИТ АДРЕС, БЛИЖАЙШИЙ;
;                          К БАЗОВОМУ                                      ;
;                       ИНАЧЕ                                              ;
;                         ФЛАГ ПЕРЕНОСА = 1                                ;
;                                                                          ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF,B,DE,HL                                  ;
;                                                                          ;
;       ВРЕМЯ:          ПРИБЛИЗИТЕЛЬНО ОТ 37 ДО 72 ТАКТОВ НА БАЙТ          ;
;                       ПЛЮС 36 ТАКТОВ ДЛЯ 8080                            ;
;                       ПРИБЛИЗИТЕЛЬНО ОТ 34 ДО 63 ТАКТОВ НА БАЙТ          ;
;                       ПЛЮС 36 ТАКТОВ ДЛЯ 8085                            ;
;                                                                          ;
;       РАЗМЕР:         ПРОГРАММА - 25 БАЙТА                               ;
;                                                                          ;
;                                                                          ;

MINELM:
        ;ЕСЛИ НЕТ ЭЛЕМЕНТОВ В МАССИВЕ, ТО ВЫЙТИ С УСТАНОВЛЕННЫМ
        ; ФЛАГОМ ПЕРЕНОСА
        MOV     A,B                     ;ПРОВЕРИТЬ РАЗМЕР МАССИВА
        ORA     A
        STC                             ;УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА, ЧТОБЫ
                                        ; УКАЗАТЬ НА ВЫХОД ПО ОШИБКЕ
        RZ                              ;ВЕРНУТЬСЯ, ЕСЛИ НЕТ ЭЛЕМЕНТОВ

        ;ЗАМЕНИТЬ ПРЕДЫДУЩИЙ ЭЛЕМЕНТ, СЧИТАВШИЙСЯ НАИМЕНЬШИМ,
        ; ТЕКУЩИМ. ПЕРВЫЙ РАЗ НАИМЕНЬШИМ СЧИТАЕТСЯ ПЕРВЫЙ ЭЛЕМЕНТ
MINLP:  MOV     A,M                     ;НАИМЕНЬШИЙ = ТЕКУЩИЙ ЭЛЕМЕНТ
        MOV     E,L                     ;СОХРАНИТЬ ЕГО АДРЕС
        MOV     D,H

        ;СРАВНИТЬ ТЕКУЩИЙ ЭЛЕМЕНТ С НАИМЕНЬШИМ
        ;ПРОДОЛЖАТЬ СРАВНЕНИЕ, ПОКА ТЕКУЩИЙ ЭЛЕМЕНТ МЕНЬШЕ
MINLP1:
        DCR     B
        JZ      EXIT
        INX     H
        CMP     M                       ;СРАВНИТЬ ТЕКУЩИЙ ЭЛЕМЕНТ С НАИМЕНЬШИМ
        JC      MINLP1                  ;ПРОДОЛЖАТЬ, ПОКА ТЕКУЩИЙ ЭЛЕМЕНТ БОЛЬШЕ
        JZ      MINLP1                  ; ИЛИ РАВЕН
        JMP     MINLP                   ;ИНАЧЕ ЗАМЕНИТЬ НАИМЕНЬШИЙ ЭЛЕМЕНТ
```

```
EXIT:
        ORA     A               ;ОЧИСТИТЬ ФЛАГ ПЕРЕНОСА - НЕТ ОШИБКИ
        XCHG                    ;HL = АДРЕС НАИМЕНЬШЕГО ЭЛЕМЕНТА
        RET


;                                                                       ;
;                                                                       ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                               ;
;                                                                       ;
;                                                                       ;

SC9D:
        LXI     H,ARY           ;HL = БАЗОВЫЙ АДРЕС МАССИВА
        MVI     B,SZARY         ;B = РАЗМЕР МАССИВА В БАЙТАХ
        CALL    MINELM
                                ;РЕЗУЛЬТАТ ДЛЯ ТЕСТОВЫХ ДАННЫХ:
                                ; A = 1 (МИНИМАЛЬНЫЙ ЭЛЕМЕНТ),
                                ; HL = АДРЕС 1 В МАССИВЕ

        JMP     SC9D            ;ЦИКЛ ДЛЯ СЛЕДУЮЩИХ ТЕСТОВ

SZARY   EQU     10H             ;РАЗМЕР МАССИВА В БАЙТАХ
ARY:    DB      8
        DB      7
        DB      6
        DB      5
        DB      4
        DB      3
        DB      2
        DB      1
        DB      0FFH
        DB      0FEH
        DB      0FDH
        DB      0FCH
        DB      0FBH
        DB      0FAH
        DB      0F9H
        DB      0F8H

        END
```

9E. ДВОИЧНЫЙ ПОИСК (BINSCH)

Ищется заданное значение в массиве беззнаковых элементов длиной в байт. Считается, что элементы расположены в возрастающем порядке. Если элемент найден, то флаг переноса очищается, в противном случае он устанавливается в 1. Возвращается адрес найденного значения. Размер массива задается; максимальное значение размера 255 байт.

Процедура. Выполняется двоичный поиск при сравнении раз за разом искомого значения с элементом, который принят за средний. После каждого сравнения отбрасывается та часть массива, которая не может содержать искомого значения (за счет того, что элементы упорядочены). Сохраняются адреса верхней и нижней границ той части, в которой производится поиск. Если значение больше среднего элемента, то отбрасывается середина и все, что на-

ходится ниже. Новой нижней границей является адрес среднего элемента плюс 1. Если искомое значение меньше среднего элемента, то отбрасывается середина и все, что находится выше ее. Новой верхней границей является адрес среднего элемента минус 1. Выход из программы осуществляется в том случае, если находится значение или если ничего не остается для поиска.

Например допустим, что есть массив 01_{16}, 02_{16}, 05_{16}, 07_{16}, 09_{16}, 09_{16}, $0D_{16}$, 10_{16}, $2E_{16}$, 37_{16}, $5D_{16}$, $7E_{16}$, $A1_{16}$, $B4_{16}$, $D7_{16}$, $E0_{16}$, и значение, которое должно быть найдено, равно $0D_{16}$. Процедура выполняется следующим образом.

При первой итерации нижней границей является базовый адрес, в верхней – адрес среднего элемента. Таким образом, результат

АДРЕС НИЖНЕЙ ГРАНИЦЫ = БАЗОВЫЙ АДРЕС
АДРЕС ВЕРХНЕЙ ГРАНИЦЫ = БАЗОВЫЙ АДРЕС + РАЗМЕР – 1 = = БАЗОВЫЙ АДРЕС + $0F_{16}$
АДРЕС СЕРЕДИНЫ = (АДРЕС ВЕРХНЕЙ ГРАНИЦЫ + АДРЕС НИЖНЕЙ ГРАНИЦЫ)/2 (результат округляется с недостатком) = БАЗОВЫЙ АДРЕС + 7
(АДРЕС СЕРЕДИНЫ) = МАССИВ(7) = 10_{16} = 16_{10}.

Так как значение ($0D_{16}$) меньше значения элемента МАССИВ(7), то элементы выше 6-го могут быть отброшены.
Результатом является

АДРЕС НИЖНЕЙ ГРАНИЦЫ = БАЗОВЫЙ АДРЕС
АДРЕС ВЕРХНЕЙ ГРАНИЦЫ = АДРЕС СЕРЕДИНЫ – 1 = БАЗОВЫЙ АДРЕС + 6
АДРЕС СЕРЕДИНЫ = (АДРЕС ВЕРХНЕЙ ГРАНИЦЫ + АДРЕС НИЖНЕЙ ГРАНИЦЫ)/2 = БАЗОВЫЙ АДРЕС + 3
(АДРЕС СЕРЕДИНЫ) = МАССИВ(3) = 07

Так как значение ($0D_{16}$) больше значения элемента МАССИВ(3), то все элементы ниже 4-го могут быть отброшены. Таким образом, результатом является

АДРЕС НИЖНЕЙ ГРАНИЦЫ = АДРЕС СЕРЕДИНЫ + 1 = БАЗОВЫЙ АДРЕС + 4
АДРЕС ВЕРХНЕЙ ГРАНИЦЫ = БАЗОВЫЙ АДРЕС + 6
АДРЕС СЕРЕДИНЫ = (АДРЕС ВЕРХНЕЙ ГРАНИЦЫ + АДРЕС НИЖНЕЙ ГРАНИЦЫ)/2 = БАЗОВЫЙ АДРЕС + 5
(АДРЕС СЕРЕДИНЫ) = МАССИВ(5) = 09

Так как значение ($0D_{16}$) больше значения элемента МАССИВ(5), то элементы ниже 6-го могут быть отброшены. Таким образом, результат равен

АДРЕС НИЖНЕЙ ГРАНИЦЫ = АДРЕС СЕРЕДИНЫ + 1 = БАЗОВЫЙ АДРЕС + 6
АДРЕС ВЕРХНЕЙ ГРАНИЦЫ = БАЗОВЫЙ АДРЕС + 6
АДРЕС СЕРЕДИНЫ = (АДРЕС ВЕРХНЕЙ ГРАНИЦЫ + АДРЕС НИЖНЕЙ ГРАНИЦЫ)/2 = БАЗОВЫЙ АДРЕС + 6
(АДРЕС СЕРЕДИНЫ) = МАССИВ(6) = $0D_{16}$

Так как значение ($0D_{16}$) равно значению элемента МАССИВ (6), то элемент найден. Если бы, с другой стороны, значение было равно $0E_{16}$, то адрес новой нижней границы был бы БАЗОВЫЙ АДРЕС + 7, и для поиска ничего бы не осталось.

УСЛОВИЯ НА ВХОДЕ

Значение для поиска в регистре А.
Размер массива в байтах в регистре С.
Базовый адрес массива (адрес наименьшего беззнакового элемента) в регистрах Н и L.

УСЛОВИЯ НА ВЫХОДЕ

Флаг переноса = 0, если значение найдено, и 1, если не найдено.
Если значение найдено, (Н и L) = его адрес.

ПРИМЕРЫ

Длина массива = 10_{16}.
Элементы массива: 01_{16}, 02_{16}, 05_{16}, 07_{16}, 09_{16}, $0D_{16}$, 10_{16}, $2E_{16}$, 37_{16}, $5D_{16}$, $7E_{16}$, $A1_{16}$, $B4_{16}$, $D7_{16}$, $E0_{16}$.

1. Данные: значение, которое должно быть найдено = $0D_{16}$.
   Результат: флаг переноса = 0, указывая тем самым, что значение найдено, (Н и L) = БАЗА + 6 (адрес, содержащий $0D_{16}$).
2. Данные: значение, которое должно быть найдено = $9B_{16}$.
   Результат: флаг переноса = 1, указывая тем самым, что значение не найдено.

```
;                                                                     ;
;                                                                     ;
;                                                                     ;
;                                                                     ;
;       ЗАГОЛОВОК:      ДВОИЧНЫЙ ПОИСК                                ;
;       ИМЯ:            BINSCH                                        ;
;                                                                     ;
;                                                                     ;


;                                                                     ;
;       НАЗНАЧЕНИЕ:     ИЩЕТ ЗАДАННОЕ ЗНАЧЕНИЕ В УПОРЯДОЧЕННОМ МАССИВЕ ;
;                       БЕЗЗНАКОВЫХ ЭЛЕМЕНТОВ ДЛИНОЙ 1 БАЙТ.           ;
;                       МАКСИМАЛЬНАЯ ДЛИНА МАССИВА 255 ЭЛЕМЕНТОВ       ;
;                                                                     ;
;       ВХОД:           ПАРА РЕГИСТРОВ Н = БАЗОВЫЙ АДРЕС МАССИВА       ;
;                       РЕГИСТР С = РАЗМЕР МАССИВА                    ;
;                       РЕГИСТР А = БАЙТ, КОТОРЫЙ ДОЛЖЕН БЫТЬ НАЙДЕН   ;
;       ВЫХОД:          ЕСЛИ ЗНАЧЕНИЕ НАЙДЕНО, ТО                     ;
;                         ФЛАГ ПЕРЕНОСА = 0                           ;
;                         ПАРА РЕГИСТРОВ Н = АДРЕС ЗНАЧЕНИЯ           ;
;                       ИНАЧЕ                                         ;
;                         ФЛАГ ПЕРЕНОСА = 1                           ;
;                                                                     ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ                                    ;
;                                                                     ;
;       ВРЕМЯ:          ПРИБЛИЗИТЕЛЬНО 123 ТАКТА НА КАЖДУЮ ИТЕРАЦИЮ    ;
;                       ПОИСКА ПЛЮС 57 ТАКТОВ ДЛЯ 8080                ;
;                       ПРИБЛИЗИТЕЛЬНО 119 ТАКТОВ НА КАЖДУЮ ИТЕРАЦИЮ   ;
;                       ПОИСКА ПЛЮС 53 ТАКТА ДЛЯ 8085                 ;
```

```
;                                                                 ;
;                    ДВОИЧНЫЙ ПОИСК ВЫПОЛНЯЕТ ПРИМЕРНО LOG N ПО   ;
;                    ОСНОВАНИЮ 2 ИТЕРАЦИЙ, ГДЕ N - ЧИСЛО ЭЛЕМЕНТОВ ;
;                    В МАССИВЕ                                    ;
;                                                                 ;
;       РАЗМЕР:      ПРОГРАММА - 42 БАЙТА                         ;
;                                                                 ;
;                                                                 ;

BINSCH:
        ;ЕСЛИ В МАССИВЕ НЕТ ЭЛЕМЕНТОВ, ТО ВЫЙТИ ИЗ ПОДПРОГРАММЫ
        ; С УСТАНОВЛЕННЫМ ФЛАГОМ ПЕРЕНОСА
        INR     C               ;ПРОВЕРИТЬ РАЗМЕР МАССИВА
        DCR     C
        STC                     ;УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА НА ТОТ СЛУЧАЙ,
                                ; ЕСЛИ РАЗМЕР МАССИВА 0
        RZ                      ;ЕСЛИ РАЗМЕР МАССИВА 0, ТО ВОЗВРАТИТЬСЯ,
                                ; УКАЗАВ, ЧТО ЗНАЧЕНИЕ НЕ НАЙДЕНО

        ;ЗАДАТЬ НАЧАЛЬНЫЕ ЗНАЧЕНИЯ АДРЕСОВ НИЖНЕЙ И ВЕРХНЕЙ ГРАНИЦ
        ; ОБЛАСТИ ПОИСКА
        ;АДРЕС НИЖНЕЙ ГРАНИЦЫ (DE) = БАЗОВЫЙ АДРЕС
        ;АДРЕС ВЕРХНЕЙ ГРАНИЦЫ (HL) = АДРЕС ПОСЛЕДНЕГО ЭЛЕМЕНТА =
        ; БАЗОВЫЙ АДРЕС + РАЗМЕР - 1
        MOV     E,L             ;АДРЕС НИЖНЕЙ ГРАНИЦЫ = БАЗОВЫЙ АДРЕС
        MOV     D,H
        MVI     B,0             ;РАСШИРИТЬ РАЗМЕР ДО 16 РАЗРЯДОВ
        DAD     B               ;АДРЕС ВЕРХНЕЙ ГРАНИЦЫ = БАЗА +
        DCX     H               ; РАЗМЕР - 1
        ;СОХРАНИТЬ ИСКОМОЕ ЗНАЧЕНИЕ
        MOV     C,A             ;СОХРАНИТЬ ИСКОМОЕ ЗНАЧЕНИЕ

        ;ИТЕРАЦИЯ ДВОИЧНОГО ПОИСКА:
        ; 1. СРАВНИТЬ ИСКОМОЕ ЗНАЧЕНИЕ СО СРЕДНИМ ЭЛЕМЕНТОМ
        ; 2. ЕСЛИ ОНИ НЕ РАВНЫ, ТО ОТБРОСИТЬ ПОЛОВИНУ, КОТОРАЯ НЕ МОЖЕТ
        ;    СОДЕРЖАТЬ ИСКОМОГО ЗНАЧЕНИЯ (ПОТОМУ ЧТО ЭТИ ЗНАЧЕНИЯ
        ;    УПОРЯДОЧЕНЫ)
        ; 3. ПРОДОЛЖАТЬ, ЕСЛИ СЛЕВА ЕСТЬ ЕЩЕ ДАННЫЕ ДЛЯ ПОИСКА
LOOP:
        ;HL = АДРЕС ВЕРХНЕЙ ГРАНИЦЫ
        ;DE = АДРЕС НИЖНЕЙ ГРАНИЦЫ
        ;C = ИСКОМОЕ ЗНАЧЕНИЕ
        ;НАЙТИ СРЕДНИЙ ЭЛЕМЕНТ
        ;АДРЕС СРЕДНЕГО ЭЛЕМЕНТА = (АДРЕС ВЕРХНЕЙ ГРАНИЦЫ +
        ; АДРЕС НИЖНЕЙ ГРАНИЦЫ) / 2
        PUSH    H               ;СОХРАНИТЬ АДРЕС ВЕРХНЕЙ ГРАНИЦЫ
        DAD     D               ;СЛОЖИТЬ АДРЕСА ВЕРХНЕЙ И НИЖНЕЙ ГРАНИЦ
        MOV     A,H             ;РАЗДЕЛИТЬ 17-РАЗРЯДНУЮ СУММУ НА 2
        RAR
        MOV     H,A
        MOV     A,L
        RAR
        MOV     L,A
        MOV     A,M             ;ВЗЯТЬ СРЕДНИЙ ЭЛЕМЕНТ

        ;СРАВНИТЬ СРЕДНИЙ ЭЛЕМЕНТ И ИСКОМОЕ ЗНАЧЕНИЕ
        CMP     C               ;СРАВНИТЬ СРЕДНИЙ ЭЛЕМЕНТ И ИСКОМОЕ
                                ; ЗНАЧЕНИЕ
```

```
        JNC     TOOLRG              ;ПЕРЕЙТИ, ЕСЛИ СРЕДНИЙ ЭЛЕМЕНТ
                                    ; РАВЕН ИЛИ БОЛЬШЕ

        ;СРЕДНИЙ ЭЛЕМЕНТ МЕНЬШЕ, ЧЕМ ИСКОМОЕ ЗНАЧЕНИЕ,
        ; ПОЭТОМУ ИЗМЕНИТЬ АДРЕС НИЖНЕЙ ГРАНИЦЫ НА АДРЕС СРЕДНЕГО
        ; ЭЛЕМЕНТА + 1, ТАК КАК ВСЕ, ЧТО НАХОДИТСЯ ВЫШЕ, ЗАВЕДОМО МЕНЬШЕ
        XCHG                        ;АДРЕС НИЖНЕЙ ГРАНИЦЫ =
        INX     D                   ;АДРЕС СРЕДНЕГО ЭЛЕМЕНТА + 1
        POP     H                   ;ВОССТАНОВИТЬ АДРЕС ВЕРХНЕЙ ГРАНИЦЫ
        JMP     CONT

        ;СРЕДНИЙ ЭЛЕМЕНТ БОЛЬШЕ ИЛИ РАВЕН ИСКОМОМУ ЗНАЧЕНИЮ,
        ; ПОЭТОМУ ИЗМЕНИТЬ АДРЕС ВЕРХНЕЙ ГРАНИЦЫ НА АДРЕС СРЕДНЕГО
        ; ЭЛЕМЕНТА - 1, ТАК КАК ВСЕ, ЧТО НАХОДИТСЯ ВЫШЕ, ЗАВЕДОМО БОЛЬШЕ
        ;ЕСЛИ ЗНАЧЕНИЕ НАЙДЕНО, ВЫЙТИ С ОЧИЩЕННЫМ ФЛАГОМ ПЕРЕНОСА
TOOLRG:
        INX     SP                  ;УДАЛИТЬ ИЗ СТЕКА АДРЕС СТАРОЙ ВЕРХНЕЙ
        INX     SP                  ; ГРАНИЦЫ
        RZ                          ;ЕСЛИ СРЕДНИЙ ЭЛЕМЕНТ РАВЕН ИСКОМОМУ
                                    ; ЗНАЧЕНИЮ, ТО ВОЗВРАТИТЬСЯ С ОЧИЩЕННЫМ
                                    ; ФЛАГОМ ПЕРЕНОСА И HL, РАВНЫМ АДРЕСУ
                                    ; ЭТОГО ЭЛЕМЕНТА
        DCX     H                   ;АДРЕС ВЕРХНЕЙ ГРАНИЦЫ =
                                    ; АДРЕС СРЕДНЕГО ЭЛЕМЕНТА - 1

        ;ПРОДОЛЖИТЬ, ЕСЛИ СЛЕВА ЕЩЕ ЕСТЬ ДАННЫЕ ДЛЯ ПОИСКА
        ;ЕСЛИ НИЖНЯЯ ГРАНИЦА ВЫШЕ ВЕРХНЕЙ ГРАНИЦЫ, ТО ДАННЫХ ДЛЯ ПОИСКА
        ; НЕТ
CONT:
        MOV     A,L                 ;СФОРМИРОВАТЬ РАЗНОСТЬ АДРЕСОВ ВЕРХНЕЙ
        CMP     E                   ; И НИЖНЕЙ ГРАНИЦ
        MOV     A,H
        SBB     D
        JNC     LOOP                ;ПРОДОЛЖИТЬ, ЕСЛИ ЕСТЬ ЕЩЕ ДАННЫЕ
                                    ; ДЛЯ ПОИСКА

        ;СЛЕВА ДАННЫХ ДЛЯ ПОИСКА НЕ ОСТАЛОСЬ, ПОЭТОМУ ЗНАЧЕНИЕ НЕ МОЖЕТ
        ; БЫТЬ НАЙДЕНО
        ;ВОЗВРАТИТЬСЯ С УСТАНОВЛЕННЫМ ФЛАГОМ ПЕРЕНОСА (ФЛАГ ДОЛЖЕН
        ; БЫТЬ УСТАНОВЛЕН, ИНАЧЕ БЫЛ БЫ ПЕРЕХОД В КОМАНДЕ JNC)
        RET


;                                                                    ;
;                                                                    ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                            ;
;                                                                    ;
;                                                                    ;

SC9E:
        ;ПОИСК ЗНАЧЕНИЯ, КОТОРОЕ ЕСТЬ В МАССИВЕ
        LXI     H,BF                ;HL = БАЗОВЫЙ АДРЕС МАССИВА
        LDA     BFSZ
        MOV     C,A                 ;C = РАЗМЕР МАССИВА В БАЙТАХ
        MVI     A,7                 ;A = ЗНАЧЕНИЕ, КОТОРОЕ ТРЕБУЕТСЯ НАЙТИ
        CALL    BINSCH              ;ПОИСК
                                    ;ФЛАГ ПЕРЕНОСА = 0 (ЗНАЧЕНИЕ НАЙДЕНО)
                                    ; HL = АДРЕС 7 В МАССИВЕ
```

```
        ;ПОИСК ЗНАЧЕНИЯ, КОТОРОГО НЕТ В МАССИВЕ
        LXI     H,BF            ;HL = БАЗОВЫЙ АДРЕС МАССИВА
        LDA     BFSZ
        MOV     C,A             ;C = РАЗМЕР МАССИВА В БАЙТАХ
        MVI     A,0             ;A = ЗНАЧЕНИЕ, КОТОРОЕ ТРЕБУЕТСЯ НАЙТИ
        CALL    BINSCH          ;ПОИСК
                                ;ФЛАГ ПЕРЕНОСА = 1 (ЗНАЧЕНИЕ НЕ НАЙДЕНО)

        JMP     SC9E            ;ЦИКЛ ДЛЯ СЛЕДУЮЩИХ ТЕСТОВ

        ;ДАННЫЕ
SIZE    EQU     010H            ;РАЗМЕР МАССИВА В БАЙТАХ
BFSZ:   DB      SIZE            ;РАЗМЕР МАССИВА В БАЙТАХ

BF:     DB      1               ;БУФЕР
        DB      2
        DB      4
        DB      5
        DB      7
        DB      9
        DB      10
        DB      11
        DB      23
        DB      50
        DB      81
        DB      123
        DB      191
        DB      199
        DB      250
        DB      255

        END
```

9F. БЫСТРАЯ СОРТИРОВКА (QSORT)

Массив беззнаковых элементов длиной в слово располагается в возрастающем порядке с использованием алгоритма быстрой сортировки. При каждой итерации выбирается некоторый элемент и массив делится на две части, одна из которых содержит элементы, большие выбранного элемента, а вторая – меньшие. Элементы, равные выбранному элементу, могут оставаться в любой части. Затем эти части тем же способом рекурсивно сортируются. Алгоритм выполняется до тех пор, пока части не содержат элементов или содержат только один элемент. Альтернативой является остановка рекурсии, когда часть содержит слишком мало элементов (скажем, меньше 20), для выполнения сортировки пузырьковым методом.

Параметрами являются базовый адрес массива, адрес его последнего элемента и наименьший доступный адрес стека. Таким образом, массив может занимать всю доступную память, т. е. столько, сколько есть места для стека. Так как процедуры получения выбранного элемента, состоящие в сравнении элементов, передвижении по массиву вперед и назад и в перестановке элементов, выполнены в виде подпрограмм, их можно легко изменить для работы с элементами других типов.

Теоретически при каждой итерации быстрой сортировки массив должен делиться пополам. То, насколько близко процедура приближается к этому

идеалу, зависит от того, насколько хорошо выбран элемент. Так как этот элемент является средней или опорной точкой, то лучшим выбором было бы центральное значение или медиана. Разумеется, действительная медиана неизвестна. Простым, но достаточно рациональным приближением является выбор медианы первого, среднего и последнего элементов.

Процедура. Сначала обрабатывается весь массив. В качестве центрального элемента выбираются медианы первого, среднего и последнего элементов. Этот элемент пересылается в первую позицию и массив делится на две части, или на два раздела. Затем рекурсивно обрабатываются эти части, которые делятся на части, и останов происходит, когда часть не содержит элементов или же содержит только один элемент. Так как при каждой рекурсии в стек помещаются 6 байт, то стек должен предохраняться от переполнения, для чего проверяется, достиг ли он своего наименьшего разрешенного положения.

Заметим, что после итерации выбранный элемент всегда остается в своем правильном месте. Следовательно, он не должен включаться в какой-либо раздел.

Если принять, что первый элемент имеет номер 1, правила для выбора среднего элемента будут следующие:

1. Если массив содержит нечетное число элементов, то взять один элемент в качестве центрального. Например, если массив содержит 11 элементов, взять элемент номер 6.

2. Если массив содержит четное число элементов и его базовый адрес четный, то взять элемент из нижней (ближе к базовому адресу) части относительно центрального. Например, если массив начинается с 300_{16} и содержит 12 элементов, взять элемент номер 6.

3. Если массив содержит четное число элементов, а его базовый адрес нечетный, взять элемент из верхней части относительно центрального. Например, если массив начинается с 301_{16} и содержит 12 элементов, взять элемент номер 7.

Используемые регистры: все.

Время выполнения: приблизительно $N * \log_2 N$ проходов в цикле PARTLP плюс дополнительно $2 * N + 1$ обращений к подпрограмме SORT. Каждая итерация PARTLP занимает приблизительно 200 тактов, а каждое обращение к SORT – приблизительно 300 тактов. Таким образом, общее время выполнения составляет примерно $200 * N * \log_2 N + 300 * (2 * N + 1)$.

Размер программы: 225 байт.

Память, необходимая для данных: 8 байт в любом месте ОЗУ для указателей на первый и последний элементы в разделе (по 2 байта, начиная соответственно с адресов FIRST и LAST), указателя на нижнюю ячейку стека (2 байта, начиная с адреса STKBTM) и исходного значения указателя стека (2 байта, начиная с адреса OLDSP).

Специальный случай: если стек переполняется (т. е. слишком близко приближается к своей границе), осуществляется выход из программы с флагом переноса, установленным в 1.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес массива в регистрах H и L.

Адрес последнего слова массива в регистрах D и E.
Наименьший доступный адрес стека в регистрах B и C.

УСЛОВИЯ НА ВЫХОДЕ

Массив отсортирован в возрастающем порядке, при этом считается, что элементами являются слова без знака. Таким образом, самое наименьшее без учета знака слово запоминается по базовому адресу. Флаг переноса = 0, если стек не переполняется и результат правильный. Флаг переноса = 1, если стек переполняется, при этом окончательный массив не отсортирован.

ПРИМЕР

1. Данные: длина (размер) массива = $0C_{16}$, элементы = $2B_{16}$, 57_{16} $1D_{16}$, 26_{16}, 22_{16}, $2E_{16}$, $0C_{16}$, 44_{16}, 17_{16}, $4B_{16}$, 37_{16}, 27_{16}.

Результат: на первой итерации: выбранный элемент – медиана первого (номер 1 = = $2B_{16}$, среднего (номер 6 = $2E_{16}$) и последнего (номер 12 = 27_{16}) элементов. Следовательно, выбирается элемент номер 1 ($2B_{16}$), и нет необходимости переставлять элементы, так как он уже находится в первой позиции.

В конце итерации массив имеет вид: 27_{16}, 17_{16}, $1D_{16}$, 26_{16}, 22_{16}, $0C_{16}$, $2B_{16}$, 44_{16}, $2E_{16}$, $4B_{16}$, 37_{16}, 57_{16}.

Первый раздел содержит элементы, меньшие $2B_{16}$: 27_{16}, 17_{16}, $1D_{16}$, 26_{16}, 22_{16} и $0C_{16}$.

Второй раздел содержит элементы, большие $2B_{16}$: 44_{16}, $2E_{16}$, $4B_{16}$, 37_{16} и 57_{16}. Заметим, что выбранный элемент ($2B_{16}$) находится теперь в нужной позиции и нет необходимости включать его в какой-либо раздел.

Теперь первый раздел может быть рекурсивно отсортирован тем же способом: выбранный элемент = медиана первого (номер 1 = 27_{16}), среднего (номер 3 = $1D_{16}$) и последнего (номер 7 = $0C_{16}$) элементов. Здесь элемент номер 4 является медианой и его необходимо сначала обменять с элементом номер 1.

Окончательным порядком элементов в первом разделе будет: $0C_{16}$, 17_{16}, $1D_{16}$, 26_{16}, 22_{16}, 27_{16}.

Первый раздел первого раздела (содержащий элементы, меньшие $1D_{16}$) содержит $0C_{16}$ и 17_{16}. Для краткости будем называть этот раздел (1,1). Второй раздел первого раздела, (содержащий элементы, большие чем $1D_{16}$): 26_{16}, 22_{16} и 27_{16}.

Как и при первой итерации, выбранный элемент находится в правильной позиции, и далее его рассматривать не надо.

Теперь раздел (1, 1) может быть рекурсивно отсортирован следующим образом: выбранный элемент = медиана первого (номер 1 = $0C_{16}$), среднего (номер 1 = $0C_{16}$) и последнего (номер 1 = 17_{16}) элементов. Очевидно, что окончательный порядок будет тот же самый, что и исходный, и два результирующих раздела содержат соответственно 0 элементов и 1 элемент. Таким образом, следующая итерация включает рекурсию, и затем тем же самым способом сортируются другие разделы. Очевидно, что когда размеры массива малы, быстрая сортировка требует больших накладных расходов. Заметим, что массив в примере не содержит одинаковых элементов. Элементы, равные выбранному элементу, во время итерации никогда не перемещаются. Таким образом, они могут оставаться в любом разделе. Точнее говоря, два раздела содержат элементы, "меньшие выбранного элемента и, возможно, равные ему" и элементы, "большие выбранного элемента, и, возможно, равные ему".

```
;                                                                        ;
;                                                                        ;
;                                                                        ;
;                                                                        ;
;       ЗАГОЛОВОК:        БЫСТРАЯ СОРТИРОВКА                             ;
;       ИМЯ:              QSORT                                          ;
;                                                                        ;
;                                                                        ;


;                                                                        ;
;       НАЗНАЧЕНИЕ:       РАСПОЛАГАЕТ ЭЛЕМЕНТЫ МАССИВА БЕЗЗНАКОВЫХ СЛОВ В ;
;                         ВОЗРАСТАЮЩЕМ ПОРЯДКЕ, ИСПОЛЬЗУЯ БЫСТРУЮ        ;
;                         СОРТИРОВКУ; МАКСИМАЛЬНАЯ ДЛИНА МАССИВА - 32767 ;
;                         СЛОВ                                           ;
;                                                                        ;
;       ВХОД:             ПАРА РЕГИСТРОВ H = АДРЕС ПЕРВОГО СЛОВА         ;
;                                            В МАССИВЕ                   ;
;                         ПАРА РЕГИСТРОВ D = АДРЕС ПОСЛЕДНЕГО СЛОВА      ;
;                                            В МАССИВЕ                   ;
;                         ПАРА РЕГИСТРОВ B = НАИМЕНЬШИЙ ДОСТУПНЫЙ АДРЕС  ;
;                                            СТЕКА                       ;
;                                                                        ;
;       ВЫХОД:            ЕСЛИ СТЕК НЕ ПЕРЕПОЛНЕН, ТО                    ;
;                           МАССИВ СОРТИРУЕТСЯ В ВОЗРАСТАЮЩЕМ ПОРЯДКЕ    ;
;                           ФЛАГ ПЕРЕНОСА = 0                            ;
;                         ИНАЧЕ                                          ;
;                           ФЛАГ ПЕРЕНОСА = 1                            ;
;                                                                        ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ                                       ;
;                                                                        ;
;       ВРЕМЯ:            ВРЕМЯ РАБОТЫ В ЗНАЧИТЕЛЬНОЙ СТЕПЕНИ ЗАВИСИТ ОТ ;
;                         ДАННЫХ, ОДНАКО ПРИ АЛГОРИТМЕ БЫСТРОЙ СОРТИРОВКИ ;
;                         ВЫПОЛНЯЕТСЯ ПРИБЛИЗИТЕЛЬНО N * (ЛОГАРИФМ N ПО  ;
;                         ОСНОВАНИЮ 2) ПРОХОДОВ В ЦИКЛЕ PARTLP. ПРИ ЭТОМ ;
;                         БУДЕТ 2 * N+1 ОБРАЩЕНИЙ К SORT. ЧИСЛО РЕКУРСИЙ, ;
;                         ВЕРОЯТНО, БУДЕТ СОСТАВЛЯТЬ НЕКОТОРУЮ ЧАСТЬ ОТ N,;
;                         НО ЕСЛИ ВСЕ ДАННЫЕ ОДИНАКОВЫЕ, ТО ЧИСЛО РЕКУРСИЙ;
;                         МОЖЕТ БЫТЬ РАВНО N. СЛЕДОВАТЕЛЬНО, РАЗМЕР СТЕКА ;
;                         ДОЛЖЕН  БЫТЬ МАКСИМАЛЬНО ВОЗМОЖНЫМ. ПРИМЕЧАНИЕ: ;
;                         КАЖДАЯ РЕКУРСИЯ ТРЕБУЕТ 6 БАЙТ В СТЕКЕ.        ;
;                                                                        ;
;                                                                        ;
;                         ПОД N ПОНИМАЕТСЯ КОЛИЧЕСТВО ЭЛЕМЕНТОВ В МАССИВЕ.;
;                                                                        ;
;       РАЗМЕР:           ПРОГРАММА - 225 БАЙТ                           ;
;                         ДАННЫЕ    - 8 БАЙТ                             ;
;                                                                        ;
;                                                                        ;

QSORT:
        ;ПРОВЕРИТЬ СТЕК НА ПЕРЕПОЛНЕНИЕ
        ;ВЫЧИСЛИТЬ ПОРОГОВЫЕ ЗНАЧЕНИЯ ДЛЯ ПРЕДУПРЕЖДЕНИЯ ПЕРЕПОЛНЕНИЯ
        ; (10 БАЙТ ОТ КОНЦА СТЕКА)
        ;СОХРАНИТЬ ЭТИ ПОРОГОВЫЕ ЗНАЧЕНИЯ ДЛЯ ПОСЛЕДУЮЩЕГО СРАВНЕНИЯ
        ;КРОМЕ ТОГО, НА ТОТ СЛУЧАЙ, ЕСЛИ ПОТРЕБУЕТСЯ АВАРИЙНО
        ; ЗАВЕРШИТЬ РАБОТУ ИЗ-ЗА ПЕРЕПОЛНЕНИЯ СТЕКА, СОХРАНИТЬ ПОЗИЦИЮ
        ; АДРЕСА ВОЗВРАТА ЭТОЙ ПОДПРОГРАММЫ
```

```
        PUSH    H               ;СОХРАНИТЬ БАЗОВЫЙ АДРЕС МАССИВА
        LXI     H,10            ;ДОБАВИТЬ МАЛЕНЬКИЙ БУФЕР (10 БАЙТ)
        DAD     B               ; К МЛАДШЕМУ АДРЕСУ СТЕКА
        SHLD    STKBTM          ;СОХРАНИТЬ СУММУ КАК КОНЕЦ СТЕКА В
                                ; КАЧЕСТВЕ КРИТЕРИЯ ДЛЯ АВАРИЙНОГО
                                ; ПРЕКРАЩАНИЯ РАБОТЫ
        LXI     H,2             ;СОХРАНИТЬ УКАЗАТЕЛЬ АДРЕСА ВОЗВРАТА
        DAD     SP              ; НА СЛУЧАЙ АВАРИЙНОГО ПРЕКРАЩЕНИЯ,
        SHLD    OLDSP           ; РАБОТЫ
        POP     H               ;ВОССТАНОВИТЬ БАЗОВЫЙ АДРЕС

        ;РЕКУРСИВНАЯ РАБОТА ПРИ АЛГОРИТМЕ БЫСТРОЙ СОРТИРОВКИ СОСТОИТ
        ; В СЛЕДУЮЩЕМ:
        ;  1. ПРОВЕРИТЬ, СОДЕРЖИТ ЛИ РАЗДЕЛ 0 ЭЛЕМЕНТОВ ИЛИ 1 ЭЛЕМЕНТ.
        ;     ЕСЛИ ЭТО ТАК, ТО ПОДНЯТЬСЯ НА ОДИН УРОВЕНЬ РЕКУРСИИ.
        ;  2. ИСПОЛЬЗУЯ МЕДИАНУ, ПОЛУЧИТЬ РАЦИОНАЛЬНОЕ ЦЕНТРАЛЬНОЕ
        ;     ЗНАЧЕНИЕ ДЛЯ ДЕЛЕНИЯ ТЕКУЩЕГО РАЗДЕЛА НА ДВЕ ЧАСТИ.
        ;  3. ПЕРЕСЫЛАТЬ ЧЕРЕЗ МАССИВ С ПОМОЩЬЮ ВЗАИМНОГО ОБМЕНА
        ;     ЭЛЕМЕНТЫ, НАРУШАЮЩИЕ ВОЗРАСТАЮЩИЙ ПОРЯДОК, ДО ТЕХ
        ;     ПОР, ПОКА ЗНАЧЕНИЯ ВСЕХ ЭЛЕМЕНТОВ, НАХОДЯЩИХСЯ НИЖЕ
        ;     ЦЕНТРАЛЬНОГО, ПРЕВЫШАЮТ ЗНАЧЕНИЯ ВСЕХ ЭЛЕМЕНТОВ, ЛЕЖАЩИХ
        ;     ВЫШЕ ЦЕНТРАЛЬНОГО. ПОДПРОГРАММА COMPARE СРАВНИВАЕТ
        ;     ЭЛЕМЕНТЫ, SWAP МЕНЯЕТ ЭЛЕМЕНТЫ МЕСТАМИ, PREV ПЕРЕДВИГАЕТ
        ;     ВЕРХНЮЮ ГРАНИЦУ ВНИЗ НА ОДИН ЭЛЕМЕНТ, А NEXT ПЕРЕСЫЛАЕТ
        ;     НИЖНЮЮ ГРАНИЦУ ВВЕРХ НА ОДИН ЭЛЕМЕНТ.
        ;  4. ПРОВЕРИТЬ СТЕК НА ПЕРЕПОЛНЕНИЕ. ЕСЛИ ЕСТЬ ПЕРЕПОЛНЕНИЕ,
        ;     АВАРИЙНО ЗАВЕРШИТЬ РАБОТУ И ВЫЙТИ.
        ;  5. УСТАНОВИТЬ ГРАНИЦЫ ДЛЯ ПЕРВОГО РАЗДЕЛА (СОДЕРЖАЩЕГО
        ;     ЭЛЕМЕНТЫ, ЗНАЧЕНИЯ КОТОРЫХ МЕНЬШЕ ЦЕНТРАЛЬНОГО)
        ;     И ОТСОРТИРОВАТЬ ИХ РЕКУРСИВНО.
        ;  6. УСТАНОВИТЬ ГРАНИЦЫ ДЛЯ ВТОРОГО РАЗДЕЛА (СОДЕРЖАЩЕГО
        ;     ЭЛЕМЕНТЫ СО ЗНАЧЕНИЯМИ, БОЛЬШИМИ ЦЕНТРАЛЬНОГО ЗНАЧЕНИЯ)
        ;     И ОТСОРТИРОВАТЬ ИХ РЕКУРСИВНО.
SORT:
        ;СОХРАНИТЬ БАЗОВЫЙ АДРЕС И АДРЕС ПОСЛЕДНЕГО ЭЛЕМЕНТА
        SHLD    FIRST           ;СОХРАНИТЬ БАЗОВЫЙ АДРЕС
                                ; В ЛОКАЛЬНОЙ ОБЛАСТИ
        XCHG
        SHLD    LAST            ;СОХРАНИТЬ АДРЕС ПОСЛЕДНЕГО ЭЛЕМЕНТА
                                ; В ЛОКАЛЬНОЙ ОБЛАСТИ

        ;ПРОВЕРИТЬ, СОДЕРЖИТ ЛИ РАЗДЕЛ 0 ЭЛЕМЕНТОВ ИЛИ 1 ЭЛЕМЕНТ.
        ; ЭТО БУДЕТ В ТОМ СЛУЧАЕ, ЕСЛИ FIRST БОЛЬШЕ LAST (0), ИЛИ
        ; РАВНО LAST (1).
PARTION:
        ;ОСТАНОВИТЬСЯ, КОГДА FIRST >= LAST
        ;DE = АДРЕС ПЕРВОГО ЭЛЕМЕНТА
        ;HL = АДРЕС ПОСЛЕДНЕГО ЭЛЕМЕНТА
        MOV     A,E             ;ВЫЧИСЛИТЬ FIRST - LAST
        SUB     L
        MOV     A,D
        SBB     H
        RNC                     ;ВОЗВРАТИТЬСЯ ИЗ ПОДПРОГРАММЫ, ЕСЛИ
                                ; РАЗНОСТЬ ПОЛОЖИТЕЛЬНАЯ. ЭТО ЗНАЧИТ,
                                ; ЧТО ДАННАЯ ЧАСТЬ ОТСОРТИРОВАНА

        ;ИСПОЛЬЗОВАТЬ МЕДИАНУ ДЛЯ НАХОЖДЕНИЯ РАЦИОНАЛЬНОГО ЦЕНТРАЛЬНОГО
        ; ЭЛЕМЕНТА. ПЕРЕСЛАТЬ ЦЕНТРАЛЬНЫЙ ЭЛЕМЕНТ В ПЕРВУЮ ПОЗИЦИЮ
```

```
        CALL    MEDIAN          ;ВЫБРАТЬ ЦЕНТРАЛЬНЫЙ ЭЛЕМЕНТ,
                                ; ПЕРЕСЛАТЬ ЕГО В ПЕРВУЮ ПОЗИЦИЮ
        MVI     C,0             ;РАЗРЯД 0 В РЕГИСТРЕ С УКАЗЫВАЕТ НА
                                ; НАПРАВЛЕНИЕ. ЕСЛИ ОН РАВЕН 0, ТО
                                ; ВВЕРХ, ИНАЧЕ - ВНИЗ

        ;ПЕРЕУПОРЯДОЧИТЬ МАССИВ, СРАВНИВАЯ С ЦЕНТРАЛЬНЫМ ЭЛЕМЕНТОМ
        ; ОСТАЛЬНЫЕ ЭЛЕМЕНТЫ. НАЧАТЬ СО СРАВНЕНИЯ С ПОСЛЕДНИМ
        ; ЭЛЕМЕНТОМ. КАЖДЫЙ РАЗ, КОГДА МЫ НАХОДИМ ЭЛЕМЕНТ,
        ; ПРИНАДЛЕЖАЩИЙ ПЕРВОЙ ЧАСТИ (Т.Е. ОН МЕНЬШЕ ЦЕНТРАЛЬНОГО
        ; ЭЛЕМЕНТА), ПЕРЕМЕЩАЕМ ЕГО С ПОМОЩЬЮ ОБМЕНА В ПЕРВУЮ ЧАСТЬ,
        ; ЕСЛИ ОН УЖЕ НЕ НАХОДИТСЯ ТАМ, А ГРАНИЦУ ПЕРВОЙ ЧАСТИ
        ; ПЕРЕМЕЩАЕМ ВНИЗ НА ОДИН ЭЛЕМЕНТ. ПОДОБНЫМ ЖЕ ОБРАЗОМ, КАЖДЫЙ
        ; РАЗ, КОГДА МЫ НАХОДИМ ЭЛЕМЕНТ, ПРИНАДЛЕЖАЩИЙ ВТОРОЙ ЧАСТИ
        ; (Т.Е. ОН БОЛЬШЕ ЦЕНТРАЛЬНОГО ЭЛЕМЕНТА), ПЕРЕМЕЩАЕМ ЕГО ВО
        ; ВТОРУЮ ЧАСТЬ, ЕСЛИ ОН УЖЕ НЕ НАХОДИТСЯ ТАМ, А ГРАНИЦУ ВТОРОЙ
        ; ЧАСТИ ПЕРЕМЕЩАЕМ ВВЕРХ НА ОДИН ЭЛЕМЕНТ.
        ;В КОНЦЕ КОНЦОВ, ГРАНИЦЫ СХОДЯТСЯ И ДЕЛЕНИЕ МАССИВА НА ЭТОМ
        ; ЗАКАНЧИВАЕТСЯ.
        ;ЗАМЕТИМ, ЧТО ЭЛЕМЕНТЫ, КОТОРЫЕ РАВНЫ ЦЕНТРАЛЬНОМУ, СОВСЕМ НЕ
        ; ПЕРЕМЕЩАЮТСЯ И МОГУТ ОСТАВАТЬСЯ В ЛЮБОЙ ЧАСТИ
        ; МАССИВА.
PARTLP:
        ;ОТСОРТИРОВАТЬ В ЦИКЛЕ НЕПРОВЕРЕННУЮ ЧАСТЬ РАЗДЕЛА ДО ТЕХ ПОР,
        ; ПОКА НИЧЕГО В НЕЙ НЕ ОСТАНЕТСЯ
        MOV     A,E             ;АДРЕС НИЖНЕЙ ГРАНИЦЫ - АДРЕС
        SUB     L               ; ВЕРХНЕЙ ГРАНИЦЫ
        MOV     A,D
        SBB     H
        JNC     DONE            ;ЕСЛИ ВСЕ ПРОВЕРЕНО, ТО ВЫЙТИ

        ;СРАВНИТЬ СЛЕДУЮЩИЕ ДВА ЭЛЕМЕНТА. ЕСЛИ ОНИ СТОЯТ НЕ ПО ПОРЯДКУ,
        ; ТО ПОМЕНЯТЬ ИХ МЕСТАМИ И ИЗМЕНИТЬ НАПРАВЛЕНИЕ ПОИСКА
        CALL    COMPARE         ;СРАВНИТЬ ЭЛЕМЕНТЫ
        JC      OK              ;ПЕРЕЙТИ, ЕСЛИ ОНИ УЖЕ РАСПОЛОЖЕНЫ
        JZ      OK              ; В ВОЗРАСТАЮЩЕМ ПОРЯДКЕ ИЛИ РАВНЫ

        ;ЭЛЕМЕНТЫ СТОЯТ НЕ ПО ПОРЯДКУ. ОБМЕНЯТЬ ИХ
        CALL    SWAP            ;ОБМЕНЯТЬ ЭЛЕМЕНТЫ
        INR     C               ;ИЗМЕНИТЬ НАПРАВЛЕНИЕ

        ;УМЕНЬШИТЬ РАЗМЕР НЕПРОВЕРЕННОЙ ОБЛАСТИ
        ;ЕСЛИ НОВЫЙ ЭЛЕМЕНТ МЕНЬШЕ ЦЕНТРАЛЬНОГО ЭЛЕМЕНТА, ТО
        ; ПЕРЕМЕСТИТЬ ВЕРХНЮЮ ГРАНИЦУ ВНИЗ
        ;ЕСЛИ НОВЫЙ ЭЛЕМЕНТ БОЛЬШЕ ЦЕНТРАЛЬНОГО ЭЛЕМЕНТА, ТО
        ; ПЕРЕМЕСТИТЬ НИЖНЮЮ ГРАНИЦУ ВВЕРХ
        ;ЕСЛИ ЭЛЕМЕНТЫ РАВНЫ, ТО ПРОДОЛЖАТЬ В СТАРОМ НАПРАВЛЕНИИ
OK:
        MOV     A,C             ;РАЗРЯД 0 РЕГИСТРА С УКАЗЫВАЕТ,
        RAR                     ; В КАКУЮ СТОРОНУ НАДО ИДТИ
        JNC     UP              ;ПЕРЕЙТИ ПРИ ДВИЖЕНИИ ВВЕРХ
        XCHG
        CALL    NEXT            ;ИНАЧЕ ПЕРЕМЕСТИТЬ ВЕРХНЮЮ ГРАНИЦУ
        XCHG                    ; ВНИЗ НА ОДИН ЭЛЕМЕНТ
        JMP     PARTLP
UP:
        CALL    PREV            ;ПЕРЕСЛАТЬ НИЖНЮЮ ГРАНИЦУ ВВЕРХ
                                ; НА ОДИН ЭЛЕМЕНТТ
```

```
        JMP     PARTLP

        ;ЭТОТ РАЗДЕЛ ТЕПЕРЬ РАЗДЕЛЕН НА ДВА РАЗДЕЛА. ОДИН НАЧИНАЕТСЯ
        ; СВЕРХУ И КОНЧАЕТСЯ ПРЯМО НАД ЦЕНТРАЛЬНЫМ ЭЛЕМЕНТОМ, ВТОРОЙ
        ; НАЧИНАЕТСЯ СРАЗУ ПОД ЦЕНТРАЛЬНЫМ ЭЛЕМЕНТОМ И ПРОДОЛЖАЕТСЯ
        ; ВНИЗ ДО КОНЦА. ЦЕНТРАЛЬНЫЙ ЭЛЕМЕНТ НАХОДИТСЯ В НУЖНОМ
        ; ОТСОРТИРОВАННОМ МЕСТЕ И ЕГО НЕ НАДО ВКЛЮЧАТЬ НИ В ОДИН РАЗДЕЛ.
DONE:
        ;СНАЧАЛА ПРОВЕРИТЬ, МОЖЕТ ЛИ СТЕК ПЕРЕПОЛНИТЬСЯ.
        ;ЕСЛИ ОН НАХОДИТСЯ СЛИШКОМ БЛИЗКО К КОНЦУ, ЗАВЕРШИТЬ АВАРИЙНО
        ; ПРОГРАММУ И ВЫЙТИ
        LXI     H,0             ;ВЫЧИСЛИТЬ STKBTM - SP
        DAD     SP
        LDA     STKBTM
        SUB     L
        LDA     STKBTM+1
        SBB     H
        JNC     ABORT           ;ВЫЙТИ, ЕСЛИ СТЕК СЛИШКОМ БОЛЬШОЙ

        ;УСТАНОВИТЬ ГРАНИЦЫ ДЛЯ ПЕРВОГО (НИЖНЕГО) РАЗДЕЛА
        ;НИЖНЯЯ ГРАНИЦА ОСТАЕТСЯ ТОЙ ЖЕ САМОЙ, ЧТО БЫЛА ДО ЭТОГО
        ;ВЕРХНЕЙ ГРАНИЦЕЙ ЯВЛЯЕТСЯ ЭЛЕМЕНТ, КОТОРЫЙ РАНЕЕ БЫЛ
        ; ЦЕНТРАЛЬНЫМ
        ;ЗАТЕМ РЕКУРСИВНО ОТСОРТИРОВАТЬ ПЕРВЫЙ РАЗДЕЛ
        PUSH    D               ;СОХРАНИТЬ АДРЕС ЦЕНТРАЛЬНОГО ЭЛЕМЕНТА
        LHLD    LAST
        PUSH    H               ;СОХРАНИТЬ АДРЕС LAST
        XCHG
        CALL    PREV            ;ВЫЧИСЛИТЬ LAST ДЛЯ ПЕРВОЙ ЧАСТИ
        XCHG
        LHLD    FIRST           ;FIRST ТОТ ЖЕ САМЫЙ, ЧТО И РАНЕЕ
        CALL    SORT            ;ОТСОРТИРОВАТЬ ПЕРВУЮ ЧАСТЬ

        ;УСТАНОВИТЬ ГРАНИЦЫ ДЛЯ ВТОРОГО (ВЕРХНЕГО) РАЗДЕЛА
        ;ВЕРХНЯЯ ГРАНИЦА ОСТАЕТСЯ ТОЙ ЖЕ САМОЙ, ЧТО БЫЛА ДО ЭТОГО
        ;НИЖНЕЙ ГРАНИЦЕЙ ЯВЛЯЕТСЯ ЭЛЕМЕНТ, КОТОРЫЙ РАНЕЕ БЫЛ
        ; ЦЕНТРАЛЬНЫМ
        ;ЗАТЕМ РЕКУРСИВНО ОТСОРТИРОВАТЬ ВТОРОЙ РАЗДЕЛ
        POP     D               ;LAST ТОТ ЖЕ САМЫЙ, ЧТО И РАНЕЕ
        POP     H               ;ВЫЧИСЛИТЬ FIRST ДЛЯ ВТОРОЙ ЧАСТИ
        CALL    NEXT
        CALL    SORT            ;ОТСОРТИРОВАТЬ ВТОРУЮ ЧАСТЬ
        ORA     A               ;ФЛАГ ПЕРЕНОСА = 0, ЧТО УКАЗЫВАЕТ
                                ; НА ОТСУТСТВИЕ ОШИБОК
        RET

        ;ВЫХОД ПО ОШИБКЕ
ABORT:  LHLD    OLDSP           ;В ВЕРШИНЕ СТЕКА НАХОДИТСЯ ИСХОДНЫЙ
        SPHL                    ; АДРЕС ВОЗВРАТА
        STC                     ;УКАЗАТЬ НА ОШИБКУ ПРИ СОРТИРОВКЕ
        RET                     ;ВЕРНУТЬСЯ В НАЧАЛЬНУЮ ВЫЗЫВАЮЩУЮ
                                ; ПРОГРАММУ

        ;*******************************
        ;ПОДПРОГРАММА: MEDIAN
        ;НАЗНАЧЕНИЕ:   ОПРЕДЕЛЯЕТ, КАКОЙ ИЗ ЭЛЕМЕНТОВ РАЗДЕЛА ДОЛЖЕН
        ;               БЫТЬ ИСПОЛЬЗОВАН В КАЧЕСТВЕ ЦЕНТРАЛЬНОГО
```

```
       ;ВХОД:  DE = АДРЕС ПЕРВОГО ЭЛЕМЕНТА (FIRST)
       ;       HL = АДРЕС ПОСЛЕДНЕГО ЭЛЕМЕНТА (LAST)
       ;ВЫХОД: DE ЯВЛЯЕТСЯ АДРЕСОМ ЦЕНТРАЛЬНОГО ЭЛЕМЕНТА
       ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF,BC,DE
       ;******************************
MEDIAN:
       ;ОПРЕДЕЛИТЬ АДРЕС СРЕДНЕГО ЭЛЕМЕНТА (MIDDLE)
       ; MIDDLE := НОРМИРОВАННОЕ(FIRST + LAST), ДЕЛЕННОЕ НА 2
       MOV     A,L             ;СЛОЖИТЬ FIRST И LAST
       ADD     E               ; БЕЗ ИЗМЕНЕНИЯ КАКОГО-ЛИБО ИЗ НИХ
       MOV     C,A             ; (НУЖНО ИСПОЛЬЗОВАТЬ 8-, А НЕ
       MOV     A,H             ; 16-РАЗРЯДНОЕ СЛОЖЕНИЕ)
       ADC     D
       RAR                     ;РАЗДЕЛИТЬ СТАРШИЙ БАЙТ СУММЫ НА 2
       MOV     B,A
       MOV     A,C
       RAR                     ;РАЗДЕЛИТЬ МЛАДШИЙ БАЙТ СУММЫ НА 2
       ANI     0FEH            ;ОЧИСТИТЬ РАЗРЯД 0 ДЛЯ НОРМИРОВАНИЯ
       MOV     C,A
       MOV     A,E             ;НОРМИРОВАТЬ MIIDLE К АДРЕСУ ГРАНИЦЫ
                               ; FIRST
       RAR                     ; ФЛАГ ПЕРЕНОСА = РАЗРЯД 0 FIRST
       MOV     A,C
       ACI     0               ;СДЕЛАТЬ РАЗРЯД 0 MIDDLE ТЕМ ЖЕ САМЫМ,
       MOV     C,A             ; ЧТО И РАЗРЯД 0 FIRST

       ;ОПРЕДЕЛИТЬ МЕДИАНУ ПЕРВОГО, СРЕДНЕГО И ПОСЛЕДНЕГО ЭЛЕМЕНТОВ
       ;СРАВНИТЬ ПЕРВЫЙ И СРЕДНИЙ ЭЛЕМЕНТЫ
       PUSH    H               ;СОХРАНИТЬ LAST
       MOV     L,C
       MOV     H,B
       CALL    COMPARE         ;СРАВНИТЬ ПЕРВЫЙ И СРЕДНИЙ ЭЛЕМЕНТЫ
       POP     H               ;ВОССТАНОВИТЬ LAST
       JNC     MIDD1           ;ПЕРЕЙТИ, ЕСЛИ ПЕРВЫЙ ЭЛЕМЕНТ >=
                               ; СРЕДНЕМУ

       ;МЫ ЗНАЕМ, ЧТО СРЕДНИЙ ЭЛЕМЕНТ > ПЕРВОГО,
       ; ПОЭТОМУ СРАВНИТЬ СРЕДНИЙ И ПОСЛЕДНИЙ ЭЛЕМЕНТЫ
       PUSH    D               ;СОХРАНИТЬ FIRST
       MOV     E,C
       MOV     D,B
       CALL    COMPARE         ;СРАВНИТЬ СРЕДНИЙ И ПОСЛЕДНИЙ ЭЛЕМЕНТЫ
       POP     D               ;ВОССТАНОВИТЬ FIRST
       JC      SWAPMF          ;ПЕРЕЙТИ, ЕСЛИ ПОСЛЕДНИЙ ЭЛЕМЕНТ >=
       JZ      SWAPMF          ; СРЕДНЕМУ; СРЕДНИЙ ЭЛЕМЕНТ ЯВЛЯЕТСЯ
                               ; МЕДИАНОЙ

       ;МЫ ЗНАЕМ, ЧТО СРЕДНИЙ ЭЛЕМЕНТ > ПЕРВОГО И СРЕДНИЙ > ПОСЛЕДНЕГО,
       ; ПОЭТОМУ СРАВНИТЬ ПЕРВЫЙ И ПОСЛЕДНИЙ ЭЛЕМЕНТЫ
       CALL    COMPARE         ;СРАВНИТЬ ПЕРВЫЙ И ПОСЛЕДНИЙ ЭЛЕМЕНТЫ
       RNC                     ;ВОЗВРАТИТЬСЯ, ЕСЛИ ПОСЛЕДНИЙ
                               ; ЭЛЕМЕНТ >= ПЕРВОМУ; ПЕРВЫЙ ЭЛЕМЕНТ
                               ; ЯВЛЯЕТСЯ МЕДИАНОЙ
       JMP     SWAPLF          ;ИНАЧЕ ПОСЛЕДНИЙ ЭЛЕМЕНТ ЯВЛЯЕТСЯ
                               ; МЕДИАНОЙ, ПОЭТОМУ ОБМЕНЯЕМ ПОСЛЕДНИЙ
                               ; И ПЕРВЫЙ ЭЛЕМЕНТЫ
```

```
        ;МЫ ЗНАЕМ, ЧТО ПЕРВЫЙ ЭЛЕМЕНТ >= СРЕДНЕМУ,
        ; ПОЭТОМУ СРАВНИТЬ ПЕРВЫЙ ЭЛЕМЕНТ И ПОСЛЕДНИЙ
MIDD1:
        CALL    COMPARE         ;СРАВНИТЬ ПОСЛЕДНИЙ И ПЕРВЫЙ ЭЛЕМЕНТЫ
        RC                      ;ВЫЙТИ, ЕСЛИ ПОСЛЕДНИЙ ЭЛЕМЕНТ >=
        RZ                      ; ПЕРВОМУ; ПЕРВЫЙ ЯВЛЯЕТСЯ МЕДИАНОЙ

        ;МЫ ЗНАЕМ, ЧТО ПЕРВЫЙ ЭЛЕМЕНТ >= СРЕДНЕМУ И ПЕРВЫЙ ЭЛЕМЕНТ >
        ; ПОСЛЕДНЕГО, ПОЭТОМУ СРАВНИТЬ СРЕДНИЙ И ПОСЛЕДНИЙ ЭЛЕМЕНТЫ
        PUSH    D               ;СОХРАНИТЬ FIRST
        MOV     E,C             ;DE = MIDDLE
        MOV     D,B
        CALL    COMPARE         ;СРАВНИТЬ СРЕДНИЙ И ПОСЛЕДНИЙ ЭЛЕМЕНТЫ
        POP     D               ;ВОССТАНОВИТЬ FIRST
        JC      SWAPLF          ;ПЕРЕЙТИ, ЕСЛИ ПОСЛЕДНИЙ ЭЛЕМЕНТ >
                                ; СРЕДНЕГО; ПОСЛЕДНИЙ ЭЛЕМЕНТ ЯВЛЯЕТСЯ
                                ; МЕДИАНОЙ

        ;СРЕДНИЙ ЭЛЕМЕНТ ЯВЛЯЕТСЯ МЕДИАНОЙ, ОБМЕНЯТЬ ЕГО С ПЕРВЫМ
SWAPMF:
        PUSH    H               ;СОХРАНИТЬ LAST
        MOV     L,C             ;HL = АДРЕС СРЕДНЕГО ЭЛЕМЕНТА
        MOV     H,B
        CALL    SWAP            ;ОБМЕНЯТЬ СРЕДНИЙ И ПЕРВЫЙ ЭЛЕМЕНТЫ
        POP     H               ;ВОССТАНОВИТЬ LAST
        RET

        ;ПОСЛЕДНИЙ ЭЛЕМЕНТ ЯВЛЯЕТСЯ МЕДИАНОЙ, ОБМЕНЯТЬ ЕГО С ПЕРВЫМ
SWAPLF:
        CALL    SWAP            ;ОБМЕНЯТЬ ПЕРВЫЙ И ПОСЛЕДНИЙ ЭЛЕМЕНТЫ
        RET

        ;******************************
        ;ПОДПРОГРАММА: NEXT
        ;НАЗНАЧЕНИЕ:   ДЕЛАЕТ HL УКАЗАТЕЛЕМ СЛЕДУЮЩЕГО ЭЛЕМЕНТА
        ;ВХОД:  HL = АДРЕС ТЕКУЩЕГО ЭЛЕМЕНТА
        ;ВЫХОД: HL = АДРЕС СЛЕДУЮЩЕГО ЭЛЕМЕНТА
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: HL
        ;******************************
NEXT:
        INX     H               ;УВЕЛИЧИТЬ ДЛЯ СЛЕДУЮЩЕГО ЭЛЕМЕНТА
        INX     H
        RET

        ;******************************
        ;ПОДПРОГРАММА: PREV
        ;НАЗНАЧЕНИЕ:   ДЕЛАЕТ HL УКАЗАТЕЛЕМ ПРЕДЫДУЩЕГО ЭЛЕМЕНТА
        ;ВХОД:  HL = АДРЕС ТЕКУЩЕГО ЭЛЕМЕНТА
        ;ВЫХОД: HL = АДРЕС ПРЕДЫДУЩЕГО ЭЛЕМЕНТА
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: HL
        ;******************************
PREV:
        DCX     H               ;УМЕНЬШИТЬ ДЛЯ ПРЕДЫДУЩЕГО ЭЛЕМЕНТА
        DCX     H
        RET

        ;******************************
```

```
        ;ПОДПРОГРАММА: COMPARE
        ;НАЗНАЧЕНИЕ:    СРАВНИВАЕТ ЭЛЕМЕНТЫ ДАННЫХ, ЗАДАННЫЕ УКАЗАТЕЛЯМИ
        ;                 В DE И HL
        ;ВХОД:   DE = АДРЕС ЭЛЕМЕНТА ДАННЫХ 1
        ;        HL = АДРЕС ЭЛЕМЕНТА ДАННЫХ 2
        ;ВЫХОД: ЕСЛИ ЭЛЕМЕНТ 1 > ЭЛЕМЕНТА 2, ТО
        ;          C = 0
        ;          Z = 0
        ;        ЕСЛИ ЭЛЕМЕНТ 1 < ЭЛЕМЕНТА 2, ТО
        ;          C = 1
        ;          Z = 0
        ;        ЕСЛИ ЭЛЕМЕНТ 1 = ЭЛЕМЕНТУ 2, ТО
        ;          C = 0
        ;          Z = 1
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF
        ;*****************************
COMPARE:
        INX     H               ;ПОЛУЧИТЬ АДРЕСА СТАРШИХ БАЙТОВ
        INX     D
        LDAX    D
        CMP     M               ;СРАВНИТЬ СТАРШИЕ БАЙТЫ
        DCX     D               ;ПОЛУЧИТЬ АДРЕСА МЛАДШИХ БАЙТОВ
        DCX     H
        RNZ                     ;ВЕРНУТЬСЯ, ЕСЛИ СТАРШИЕ БАЙТЫ НЕ РАВНЫ
        LDAX    D               ;ИНАЧЕ СРАВНИТЬ МЛАДШИЕ БАЙТЫ
        CMP     M
        RET

        ;*****************************
        ;ПОДПРОГРАММА: SWAP
        ;НАЗНАЧЕНИЕ:    ОБМЕНИВАЕТ ЭЛЕМЕНТЫ, ЗАДАННЫЕ УКАЗАТЕЛЯМИ
        ;                 В DE И HL
        ;ВХОД:   DE = АДРЕС ЭЛЕМЕНТА 1
        ;        HL = АДРЕС ЭЛЕМЕНТА 2
        ;ВЫХОД: ЭЛЕМЕНТЫ МЕНЯЮТСЯ МЕСТАМИ
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF,B
        ;*****************************

SWAP:
        ;ОБМЕНЯТЬ МЛАДШИЕ БАЙТЫ
        MOV     B,M             ;ВЗЯТЬ ЭЛЕМЕНТ 2
        LDAX    D               ;ВЗЯТЬ ЭЛЕМЕНТ 1
        MOV     M,A             ;ЗАПОМНИТЬ НОВЫЙ ЭЛЕМЕНТ 2
        MOV     A,B
        STAX    D               ;ЗАПОМНИТЬ НОВЫЙ ЭЛЕМЕНТ 1
        INX     H
        INX     D

        ;ОБМЕНЯТЬ СТАРШИЕ БАЙТЫ
        MOV     B,M             ;ВЗЯТЬ ЭЛЕМЕНТ 2
        LDAX    D               ;ВЗЯТЬ ЭЛЕМЕНТ 1
        MOV     M,A             ;ЗАПОМНИТЬ ЭЛЕМЕНТ 2
        MOV     A,B
        STAX    D               ;ЗАПОМНИТЬ ЭЛЕМЕНТ 1
        DCX     H
        DCX     D
        RET
```

```
        ;СЕКЦИЯ ДАННЫХ
FIRST:  DS      2                   ;УКАЗАТЕЛЬ НА ПЕРВЫЙ ЭЛЕМЕНТ ЧАСТИ
LAST:   DS      2                   ;УКАЗАТЕЛЬ НА ПОСЛЕДНИЙ ЭЛЕМЕНТ ЧАСТИ
STKBTM: DS      2                   ;ПОРОГОВОЕ ЗНАЧЕНИЕ ДЛЯ ПЕРЕПОЛНЕНИЯ
                                    ; СТЕКА
OLDSP:  DS      2                   ;УКАЗАТЕЛЬ НА НАЧАЛЬНЫЙ АДРЕС ВОЗВРАТА
;                                                                          ;
;                                                                          ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                                  ;
;                                                                          ;
;                                                                          ;
SC9F:
        ;ОТСОРТИРОВАТЬ МАССИВ МЕЖДУ BEGBUF (ПЕРВЫЙ ЭЛЕМЕНТ)
        ; И ENDBUF (ПОСЛЕДНИЙ ЭЛЕМЕНТ)
        ;В НАЧАЛЕ ВЕРШИНА СТЕКА НАХОДИТСЯ ПО ШЕСТНАДЦАТЕРИЧНОМУ АДРЕСУ
        ; 5000; СТЕК МОЖНО РАСШИРЯТЬ ДО ШЕСТНАДЦАТЕРИЧНОГО АДРЕСА 4F00
        LXI     SP,5000H            ;УСТАНОВИТЬ ОБЛАСТЬ СТЕКА
        LXI     B,4F00H             ;BC = НИЖНИЙ ДОСТУПНЫЙ АДРЕС СТЕКА
        LXI     H,BEGBUF            ;HL = АДРЕС ПЕРВОГО ЭЛЕМЕНТА МАССИВА
        LXI     D,ENDBUF            ;DE = АДРЕС ПОСЛЕДНЕГО ЭЛЕМЕНТА МАССИВА
        CALL    QSORT               ;ОТСОРТИРОВАТЬ
                                    ;РЕЗУЛЬТАТ ДЛЯ ТЕСТОВЫХ ДАННЫХ:
                                    ; 0,1,2,3, ... ,14,15

        JMP     SC9F                ;ЦИКЛ ДЛЯ СЛЕДУЮЩИХ ТЕСТОВ

        ;СЕКЦИЯ ДАННЫХ
BEGBUF: DW      15
        DW      14
        DW      13
        DW      12
        DW      11
        DW      10
        DW      9
        DW      8
        DW      7
        DW      6
        DW      5
        DW      4
        DW      3
        DW      2
        DW      1
ENDBUF: DW      0

        END
```

СПИСОК ЛИТЕРАТУРЫ

Augenstein M. J., Tenenbaum A. M. Data Structures and PL/1 Programming. Englewood Cliffs, N. J.: Prentice-Hall, 1979, pp. 460 – 71. Аналогичная книга для языка Паскаль издана под названием Data Structures Using Pascal (Englewood Cliffs, N. J.: Prentice-Hall, 1982).

Bowles K. L. Microcomputer Problem Solving Using Pascal, New York: Springer – Verlag, 1977.

Knuth D. E. The Art of Computer Programming. Vol. 3: Searching and Sorting. Reading, Mass.: Addison-Wesley, 1973, pp. 114 – 23.
[Имеется перевод: Кнут Д. Е. Искусство программирования. Т.3: Сортировка и поиск. – М.: Мир, 1978.]

Тестируется область ОЗУ, заданная базовым адресом и длиной в байтах. В каждый байт записывается 0, FF_{16}, AA_{16} (10101010_2) и 55_{16} (01010101_2) и проверяется, могут ли они быть правильно считаны. В каждую позицию разряда каждого байта помещается 1 и проверяется, может ли быть эта единица правильно считана при всех остальных очищенных разрядах. Если все тесты работают правильно, то очищается флаг переноса. При нахождении ошибки осуществляется немедленный выход из программы с установлением флага переноса и возвратом тестового значения и адреса, при котором произошла ошибка.

Процедура. При проверке с одним значением (с 0, FF_{16}, AA_{16} и 55_{16}) сначала заполняется область памяти, а затем каждый байт сравнивается с заданным значением. Заполнение сначала всей области памяти должно обеспечить достаточный временной разрыв между записью и чтением, чтобы определить ошибки сохранения данных (которые могут возникнуть при неправильной разработке схем обновления). Затем, начиная с разряда 7, выполняется тест с перемещающимся разрядом; здесь записываются данные в память и осуществляется попытка немедленно прочитать их назад для сравнения.

Используемые регистры: все.
Время выполнения: приблизительно 754 такта на байт, который тестируется, плюс 485 тактов (8080) или 756 тактов на тестируемый байт плюс 481 такт (8085).
Размер программы: 83 байта.
Память, необходимая для данных: отсутствует.
Специальные случаи:

1. Размер области 0000_{16} вызывает немедленный выход без проверки памяти. Флаг переноса очищается, что указывает на отсутствие ошибок.
2. Так как программа изменяет все байты тестируемой области, использование ее для проверки области, содержащей саму программу, может привести к непредсказуемым результатам. Заметим, что первый случай означает, что этой программой нельзя проверять всю память. Такой запрос был бы бессмысленным в любом случае, так как он требовал бы проверки области самой программы.
3. Проверка ПЗУ вызывает возврат с ошибкой, появляющейся сразу после того, как значение теста будет отличаться от содержимого памяти.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес тестируемой области в регистрах H и L.
Размер тестируемой области в байтах в регистрах D и E.

УСЛОВИЯ НА ВЫХОДЕ

1. Если найдена ошибка, то:
   флаг переноса = 1,
   адрес, содержащий ошибку, в регистрах H и L,
   тестовое значение в A.
2. Если не найдена ошибка, то:
   флаг переноса = 0,
   все байты в тестируемой области содержат 0.

ПРИМЕР

1. Данные: базовый адрес = 0380_{16},
длина (размер) области = 0200_{16}.

Результат: тестируемая область содержит 0200_{16} байт, начиная с адреса 0380_{16}, т. е. адреса с 0380_{16} до $057F_{16}$. Порядок проверки следующий:

1. Записать и прочитать 0.
2. Записать и прочитать FF_{16}.
3. Записать и прочитать AA_{16} (10101010_2).
4. Записать и прочитать 55_{16} (01010101_2).
5. Осуществить тест с перемещающимся разрядом, начинающийся с 1 в разряде 7. Таким образом, тест начинается с 10000000_2 (80_{16}) и для каждой последующей проверки байта пересылается на одну позицию вправо единица.

```
;                                                              ;
;                                                              ;
;       ЗАГОЛОВОК:      ТЕСТ ОЗУ                               ;
;       ИМЯ:            RAMTST                                 ;
;                                                              ;
;                                                              ;


;                                                              ;
;       НАЗНАЧЕНИЕ:     ПРОВЕРЯЕТ ОБЛАСТЬ ОЗУ (ПАМЯТЬ ЧТЕНИЯ-ЗАПИСИ) ;
;                         1. ЗАПИСЫВАЕТ ВСЕ 0 И ПРОВЕРЯЕТ      ;
;                         2. ЗАПИСЫВАЕТ ВСЕ ШЕСТНАДЦАТЕРИЧНЫЕ ЧИСЛА FF ;
;                            И ПРОВЕРЯЕТ                       ;
;                         3. ЗАПИСЫВАЕТ ВСЕ ШЕСТНАДЦАТЕРИЧНЫЕ ЧИСЛА AA ;
;                            И ПРОВЕРЯЕТ                       ;
;                         4. ЗАПИСЫВАЕТ ВСЕ ШЕСТНАДЦАТЕРИЧНЫЕ ЧИСЛА 55 ;
;                            И ПРОВЕРЯЕТ                       ;
;                         5. СДВИГАЕТ ОДНУ ЕДИНИЦУ ЧЕРЕЗ ВСЕ РАЗРЯДЫ С ;
;                            ОДНОВРЕМЕННЫМ ОЧИЩЕНИЕМ ОСТАЛЬНЫХ РАЗРЯДОВ. ;
;                                                              ;
;                         ЕСЛИ ПРОГРАММА ОБНАРУЖИВАЕТ ОШИБКУ, ТО ОНА ;
;                         НЕМЕДЛЕННО ЗАКАНЧИВАЕТ РАБОТУ С ФЛАГОМ ;
;                         ПЕРЕНОСА, УСТАНОВЛЕННЫМ В ЕДИНИЦУ; ПРИ ЭТОМ ;
;                         УКАЗЫВАЕТСЯ ТЕСТОВОЕ ЗНАЧЕНИЕ И МЕСТО ОШИБКИ. ;
;                                                              ;
;       ВХОД:           ПАРА РЕГИСТРОВ H = БАЗОВЫЙ АДРЕС ПРОВЕРЯЕМОЙ ;
;                       ОБЛАСТИ                                ;
;                       ПАРА РЕГИСТРОВ D = РАЗМЕР ОБЛАСТИ В БАЙТАХ ;
;                                                              ;
;       ВЫХОД:          ЕСЛИ НЕТ ОШИБОК, ТО                    ;
;                         ФЛАГ ПЕРЕНОСА РАВЕН 0                ;
;                         ПРОВЕРЯЕМАЯ ОБЛАСТЬ СОДЕРЖИТ НУЛИ ВО ВСЕХ ;
;                         БАЙТАХ                               ;
;                       ИНАЧЕ                                  ;
;                         ФЛАГ ПЕРЕНОСА РАВЕН 1                ;
;                         ПАРА РЕГИСТРОВ H = АДРЕС ОШИБКИ      ;
;                         РЕГИСТР A = ТЕСТОВОЕ ЗНАЧЕНИЕ        ;
;                                                              ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ                             ;
;                                                              ;
;       ВРЕМЯ:          ПРИБЛИЗИТЕЛЬНО 754 ТАКТА НА БАЙТ ПЛЮС  ;
;                       485 ТАКТОВ ДЛЯ 8080                    ;
;                       ПРИБЛИЗИТЕЛЬНО 756 ТАКТОВ НА БАЙТ ПЛЮС ;
```

```
;                     481 ТАКТ ДЛЯ 8085                              ;
;                                                                    ;
;       РАЗМЕР:       ПРОГРАММА - 83 БАЙТА                           ;
;                                                                    ;
;                                                                    ;

RAMTST:
        ;ВЫЙТИ БЕЗ ОШИБОК, ЕСЛИ РАЗМЕР ОБЛАСТИ РАВЕН 0
        MOV     A,D             ;ПРОВЕРИТЬ РАЗМЕР ОБЛАСТИ
        ORA     E
        RZ                      ;ЕСЛИ РАЗМЕР РАВЕН НУЛЮ, ТО ВЫЙТИ БЕЗ
                                ; ОШИБОК

        ;ЗАПОЛНИТЬ ПАМЯТЬ ЧИСЛОМ 0 И ПРОВЕРИТЬ
        MVI     C,0
        CALL    FILCMP
        RC                      ;ВЫЙТИ, ЕСЛИ НАЙДЕНА ОШИБКА

        ;ЗАПОЛНИТЬ ПАМЯТЬ ШЕСТНАДЦАТЕРИЧНЫМ ЧИСЛОМ FF (ВСЕ ЕДИНИЦЫ)
        ; И ПРОВЕРИТЬ
        MVI     C,0FFH
        CALL    FILCMP
        RC                      ;ВЫЙТИ, ЕСЛИ НАЙДЕНА ОШИБКА

        ;ЗАПОЛНИТЬ ПАМЯТЬ ШЕСТНАДЦАТЕРИЧНЫМ ЧИСЛОМ AA (ПЕРЕМЕЖАЮЩИЕСЯ
        ; ЕДИНИЦЫ И НУЛИ) И ПРОВЕРИТЬ
        MVI     C,0AAH
        CALL    FILCMP
        RC                      ;ВЫЙТИ, ЕСЛИ НАЙДЕНА ОШИБКА

        ;ЗАПОЛНИТЬ ПАМЯТЬ ШЕСТНАДЦАТЕРИЧНЫМ ЧИСЛОМ 55 (ПЕРЕМЕЖАЮЩИЕСЯ
        ; НУЛИ И ЕДИНИЦЫ) И ПРОВЕРИТЬ
        MVI     C,55H
        CALL    FILCMP
        RC                      ;ВЫЙТИ, ЕСЛИ НАЙДЕНА ОШИБКА

        ;ВЫПОЛНИТЬ ТЕСТ С ПЕРЕМЕЩАЮЩИМСЯ РАЗРЯДОМ. ПОМЕСТИТЬ ЕДИНИЦУ
        ; В РАЗРЯД 7 И ПОСМОТРЕТЬ, ЧИТАЕТСЯ ЛИ ОНА. ЗАТЕМ ПЕРЕСЛАТЬ
        ; ЕДИНИЦУ В РАЗРЯДЫ 6, 5, 4, 3, 2, 1 И 0 И ПОСМОТРЕТЬ,
        ; ЧИТАЕТСЯ ЛИ ОНА
WKLP:
        MVI     A,10000000B     ;СДЕЛАТЬ РАЗРЯД 7 РАВНЫМ 1, ВСЕ
                                ; ОСТАЛЬНЫЕ РАЗРЯДЫ - НУЛЕВЫМИ
WKLP1:
        MOV     M,A             ;ЗАПИСАТЬ ШАБЛОН ТЕСТА В ПАМЯТЬ
        CMP     M               ;ПОПЫТАТЬСЯ ПРОЧИТАТЬ ЕГО
        STC                     ;УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА НА СЛУЧАЙ
                                ; ОШИБКИ
        RNZ                     ;ВЫЙТИ В СЛУЧАЕ ОШИБКИ
        RRC                     ;СДВИНУТЬ ШАБЛОН ЦИКЛИЧЕСКИ, ЧТОБЫ
                                ; ПЕРЕМЕСТИТЬ 1 ВПРАВО
        CPI     10000000B
        JNZ     WKLP1           ;ПРОДОЛЖАТЬ, ПОКА 1 НЕ ВЕРНЕТСЯ
                                ; В РАЗРЯД 7
        MVI     M,0             ;ОЧИСТИТЬ УЖЕ ПРОВЕРЕННЫЙ БАЙТ
        INX     H
        DCX     D               ;УМЕНЬШИТЬ И ПРОВЕРИТЬ 16-РАЗРЯДНЫЙ
```

```
        MOV     A,D                     ; СЧЕТЧИК
        ORA     E
        JNZ     WKLP                    ;ПРОДОЛЖАТЬ ДЛЯ ВСЕЙ ПРОВЕРЯЕМОЙ ПАМЯТИ
        RET                             ;НЕТ ОШИБОК (ЗАМЕТИМ, ЧТО ORA E
                                        ; ОЧИЩАЕТ ФЛАГ ПЕРЕНОСА)

        ;*************************************
        ;ПОДПРОГРАММА: FILCMP
        ;НАЗНАЧЕНИЕ:    ЗАПОЛНЯЕТ ПАМЯТЬ ТЕСТОВЫМ ЗНАЧЕНИЕМ И ПРОВЕРЯЕТ,
        ;               МОЖЕТ ЛИ ОНО БЫТЬ ПРОЧИТАНО
        ;ВХОД:  C = ТЕСТОВОЕ ЗНАЧЕНИЕ
        ;       HL= БАЗОВЫЙ АДРЕС
        ;       DE = РАЗМЕР ОБЛАСТИ В БАЙТАХ
        ;ВЫХОД: ЕСЛИ НЕТ ОШИБОК,ТО
        ;         ФЛАГ ПЕРЕНОСА = 0
        ;       ИНАЧЕ
        ;         ФЛАГ ПЕРЕНОСА = 1
        ;         HL = АДРЕС ОШИБКИ
        ;         DE = РАЗМЕР ОБЛАСТИ В БАЙТАХ
        ;         BC = БАЗОВЫЙ АДРЕС
        ;         A = ТЕСТОВОЕ ЗНАЧЕНИЕ
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ
        ;*************************************
FILCMP:
        PUSH    H                       ;СОХРАНИТЬ БАЗОВЫЙ АДРЕС
        PUSH    D                       ;СОХРАНИТЬ РАЗМЕР ОБЛАСТИ

        ;ЗАПОЛНИТЬ ПАМЯТЬ ТЕСТОВЫМ ЗНАЧЕНИЕМ
FILLP:
        MOV     M,C                     ;ЗАПОЛНИТЬ БАЙТ ТЕСТОВЫМ ЗНАЧЕНИЕМ
        INX     H                       ;УВЕЛИЧИТЬ ДЛЯ СЛЕДУЮЩЕГО БАЙТА
        DCX     D
        MOV     A,D
        ORA     E
        JNZ     FILLP                   ;ПРОДОЛЖАТЬ, ПОКА НЕ ЗАПОЛНЕНА ВСЯ ПАМЯТЬ

        POP     D                       ;ВОССТАНОВИТЬ РАЗМЕР ОБЛАСТИ
        POP     H                       ;ВОССТАНОВИТЬ БАЗОВЫЙ АДРЕС
        PUSH    H                       ;СОХРАНИТЬ БАЗОВЫЙ АДРЕС
        PUSH    D                       ;СОХРАНИТЬ РАЗМЕР ОБЛАСТИ

        ;СРАВНИТЬ ПАМЯТЬ И ТЕСТОВОЕ ЗНАЧЕНИЕ
CMPLP:
        MOV     A,M                     ;ВЗЯТЬ ЗНАЧЕНИЕ ИЗ ПАМЯТИ
        CMP     C                       ;ОНО РАВНО ТЕСТОВОМУ ЗНАЧЕНИЮ?
        JNZ     CMPER                   ;НЕТ, ВЫЙТИ, УКАЗАВ НА ОШИБКУ
        INX     H
        DCX     D                       ;ПРОВЕРИТЬ 16-РАЗРЯДНЫЙ СЧЕТЧИК БАЙТОВ
        MOV     A,D
        ORA     E
        JNZ     CMPLP                   ;ПРОДОЛЖАТЬ ДО СРАВНЕНИЯ ВСЕХ БАЙТОВ

        ;ОШИБКА НЕ НАЙДЕНА, ФЛАГ ПЕРЕНОСА УЖЕ ОЧИЩЕН (КОМАНДОЙ ORA E)
        POP     D                       ;DE = РАЗМЕР ОБЛАСТИ В БАЙТАХ
        POP     H                       ;HL = БАЗОВЫЙ АДРЕС
        RET                             ;ВЫЙТИ (ORA E ОСТАВЛЯЕТ ФЛАГ ПЕРЕНОСА
                                        ; РАВНЫМ 0)
```

```
        ;ВЫХОД ПО ОШИБКЕ, УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА
        ;HL = АДРЕС ОШИБКИ
        ;A = ТЕСТОВОЕ ЗНАЧЕНИЕ
CMPER:
        MOV     A,C                     ;A = ТЕСТОВОЕ ЗНАЧЕНИЕ
        POP     D                       ;DE = РАЗМЕР ОБЛАСТИ В БАЙТАХ
        POP     B                       ;BC = БАЗОВЫЙ АДРЕС
        STC                             ;УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА, УКАЗЫВАЮЩИЙ
                                        ; НА ОШИБКУ
        RET



;                                                                      ;
;                                                                      ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                              ;
;                                                                      ;
;                                                                      ;

SC9G:
        ;ПРОВЕРИТЬ ОЗУ ОТ 2000  ДО 300F (АДРЕСА ШЕСТНАДЦАТЕРИЧНЫЕ)
        ; РАЗМЕР ОБЛАСТИ = 1010 (ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО) БАЙТ
        LXI     H,2000H                 ;HL = БАЗОВЫЙ АДРЕС
        LXI     D,1010H                 ;DE = ЧИСЛО БАЙТОВ
        CALL    RAMTST                  ;ПРОВЕРИТЬ ПАМЯТЬ
                                        ;ФЛАГ ПЕРЕНОСА ДОЛЖЕН БЫТЬ РАВЕН 0

        JMP     SC9G                    ;ЦИКЛ ДЛЯ СЛЕДУЮЩИХ ТЕСТОВ

        END
```

9H. ТАБЛИЦА ПЕРЕХОДОВ (JTAB)

Передается управление по адресу, выбранному из таблицы в соответствии с индексом. Адрес запоминается в обычном для 8080/8085 формате (младший по значению байт идет первым), начиная с адреса JMPTAB. Постоянная величина LENSUB содержит размер таблицы (число адресов), который должен быть меньше или равен 128. Если индекс больше или равен LENSUB, программа немедленно возвращает управление с флагом переноса, установленным в 1.

Процедура. Сначала проверяется, не больше ли индекс размера таблицы (LENSUB) или равен ему. Если это так, управление возвращается с флагом переноса, установленным в 1. Если это не так, программа получает начальный адрес соответствующей подпрограммы из таблицы и переходит к ней.

Используемые регистры: AF.
Время выполнения: 118 (8080) или 116 (8085) тактов, не считая времени на выполнение действительной подпрограммы.
Размер программы: 21 байт плюс 2 * LENSUB байт для таблицы начальных адресов, где LENSUB – число подпрограмм.
Память, необходимая для данных: отсутствует.
Специальный случай: обращение с индексом, большим или равным LENSUB, вызывает немедленный выход из программы с флагом переноса, установленным в 1.

УСЛОВИЯ НА ВХОДЕ

Индекс в регистре A.

УСЛОВИЯ НА ВЫХОДЕ

Если (A) больше, чем LENSUB – 1, происходит немедленный возврат с флагом переноса, равным 1. В противном случае управление передается соответствующей подпрограмме как при выполнении индексированного вызова. Адрес возврата остается в вершине стека.

ПРИМЕР

1. Данные: LENSUB (размер таблицы подпрограмм) = 03, таблица содержит адреса SUB0, SUB1 и SUB2. индекс = (A) = 02.

Результат: управление передается по адресу SUB2 (счетчик команд = SUB2).

```
;                                                                   ;
;                                                                   ;
;                                                                   ;
;                                                                   ;
;       ЗАГОЛОВОК:      ТАБЛИЦА ПЕРЕХОДОВ                           ;
;       ИМЯ:            JTAB                                        ;
;                                                                   ;
;                                                                   ;


;                                                                   ;
;       НАЗНАЧЕНИЕ:     ПО ЗАДАННОМУ ИНДЕКСУ ПЕРЕХОДИТ К СООТВЕТСТВУЮЩЕЙ;
;                       ЭТОМУ ИНДЕКСУ ПОДПРОГРАММЕ В ТАБЛИЦЕ        ;
;                                                                   ;
;       ВХОД:           РЕГИСТР A ЯВЛЯЕТСЯ НОМЕРОМ ПОДПРОГРАММЫ (ОТ 0 ;
;                                 ДО LENSUB-1,ГДЕ LENSUB - ЧИСЛО    ;
;                                 ПОДПРОГРАММ). LENSUB ДОЛЖНО БЫТЬ  ;
;                                 МЕНЬШЕ ИЛИ РАВНО 128.             ;
;                                                                   ;
;       ВЫХОД:          ЕСЛИ НОМЕР ПОДПРОГРАММЫ ПРАВИЛЬНЫЙ, ТО      ;
;                         ВЫПОЛНЯЕТ ПОДПРОГРАММУ                    ;
;                       ИНАЧЕ                                       ;
;                         ФЛАГ ПЕРЕНОСА = 1                         ;
;                                                                   ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ:  AF                                  ;
;                                                                   ;
;       ВРЕМЯ:          118 ТАКТОВ ПЛЮС ВРЕМЯ ВЫПОЛНЕНИЯ ПОДПРОГРАММЫ ;
;                       ДЛЯ 8080                                    ;
;                       116 ТАКТОВ ПЛЮС ВРЕМЯ ВЫПОЛНЕНИЯ ПОДПРОГРАММЫ ;
;                       ДЛЯ 8085                                    ;
;                                                                   ;
;       РАЗМЕР:         ПРОГРАММА - 21 БАЙТ ПЛЮС РАЗМЕР ТАБЛИЦЫ     ;
;                                    (2*LENSUB)                     ;
;                                                                   ;
;                                                                   ;

        ;ВЫЙТИ С УСТАНОВЛЕННЫМ ФЛАГОМ ПЕРЕНОСА, ЕСЛИ НОМЕР ПОДПРОГРАММЫ
        ;НЕПРАВИЛЬНЫЙ, Т.Е. ЕСЛИ ОН СЛИШКОМ БОЛЬШОЙ ДЛЯ ТАБЛИЦЫ
        ; (>LENSUB - 1)
JTAB:
        CPI     LENSUB          ;СРАВНИТЬ НОМЕР ПОДПРОГРАММЫ И РАЗМЕР
                                ; ТАБЛИЦЫ
```

```
        CMC                             ;ИНВЕРТИРОВАТЬ ФЛАГ ПЕРЕНОСА ДЛЯ
                                        ; ИНДИКАЦИИ ОШИБКИ
        RC                              ;ЕСЛИ НОМЕР ПОДПРОГРАММЫ СЛИШКОМ
                                        ; БОЛЬШОЙ, ТО ВЕРНУТЬСЯ С УСТАНОВЛЕННЫМ
                                        ; ФЛАГОМ
                                        ; ПЕРЕНОСА

        ;ПО ИНДЕКСУ НАЙТИ В ТАБЛИЦЕ АДРЕС ДЛИНОЙ В СЛОВО, ОСТАВЛЯЯ ПАРЫ
        ; РЕГИСТРОВ БЕЗ ИЗМЕНЕНИЯ, ЧТОБЫ МОЖНО БЫЛО ИХ ИСПОЛЬЗОВАТЬ
        ; ДЛЯ ПЕРЕДАЧИ ПАРАМЕТРОВ
        PUSH    H                       ;СОХРАНИТЬ HL
        ADD     A                       ;УДВОИТЬ ИНДЕКС ДЛЯ ЗАПИСЕЙ ДЛИНОЙ В
                                        ;СЛОВО
        LXI     H,JMPTAB                ;ИНДЕКСИРОВАТЬ В ТАБЛИЦЕ, ИСПОЛЬЗУЯ
        ADD     L                       ;  8-РАЗРЯДНОЕ СЛОЖЕНИЕ, С ТЕМ ЧТОБЫ
        MOV     L,A                     ; ИЗБЕЖАТЬ ИЗМЕНЕНИЯ ДРУГИХ ПАР РЕГИСТРОВ
        MVI     A,0
        ADC     H
        MOV     H,A                     ;АДРЕС ДОСТУПА К ПОДПРОГРАММЕ

        ;ПОЛУЧИТЬ ИЗ ТАБЛИЦЫ АДРЕС ПОДПРОГРАММЫ И ПЕРЕДАТЬ НА НЕЕ
        ; УПРАВЛЕНИЕ, ОСТАВЛЯЯ ВСЕ ПАРЫ РЕГИСТРОВ БЕЗ ИЗМЕНЕНИЯ
        MOV     A,M                     ;ПЕРЕСЛАТЬ АДРЕС ПОДПРОГРАММЫ В HL
        INX     H
        MOV     H,M
        MOV     L,A
        XTHL                            ;ВОССТАНОВИТЬ СТАРОЕ ЗНАЧЕНИЕ HL,
                                        ; ЗАГРУЗИТЬ В СТЕК АДРЕС ПОДПРОГРАММЫ
        RET                             ;ПЕРЕЙТИ НА ПОДПРОГРАММУ

LENSUB  EQU     3                       ;ЧИСЛО ПОДПРОГРАММ В ТАБЛИЦЕ

JMPTAB:                                 ;ТАБЛИЦА ПЕРЕХОДОВ
        DW      SUB0                    ;ПОДПРОГРАММА 0
        DW      SUB1                    ;ПОДПРОГРАММА 1
        DW      SUB2                    ;ПОДПРОГРАММА 2

        ;ТРИ ТЕСТОВЫЕ ПОДПРОГРАММЫ ДЛЯ ТАБЛИЦЫ ПЕРЕХОДОВ
SUB0:
        MVI     A,1                     ;ТЕСТОВАЯ ПОДПРОГРАММА 0
                                        ; УСТАНАВЛИВАЕТ (A) = 1
        RET

SUB1:
        MVI     A,2                     ;ТЕСТОВАЯ ПОДПРОГРАММА 1
                                        ; УСТАНАВЛИВАЕТ (A) = 2
        RET

SUB2:
        MVI     A,3                     ;ТЕСТОВАЯ ПОДПРОГРАММА 2
                                        ; УСТАНАВЛИВАЕТ (A) = 3
        RET


;                                                                       ;
;                                                                       ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                               ;
```

```
;                                                                       ;
;                                                                       ;

SC9H:
        SUB     A                   ;ВЫПОЛНИТЬ ПОДПРОГРАММУ 0
        CALL    JTAB                ; ПОСЛЕ ВЫПОЛНЕНИЯ (A) = 1
        MVI     A,1                 ;ВЫПОЛНИТЬ ПОДПРОГРАММУ 1
        CALL    JTAB                ; ПОСЛЕ ВЫПОЛНЕНИЯ (A) = 2
        MVI     A,2                 ;ВЫПОЛНИТЬ ПОДПРОГРАММУ 2
        CALL    JTAB                ; ПОСЛЕ ВЫПОЛНЕНИЯ (A) = 3
        MVI     A,3                 ;ВЫПОЛНИТЬ ПОДПРОГРАММУ 3
        CALL    JTAB                ; ПОСЛЕ ВЫПОЛНЕНИЯ ФЛАГ ПЕРЕНОСА = 1
        JMP     SC9H                ;ЦИКЛ ДЛЯ СЛЕДУЮЩИХ ТЕСТОВ

        END
```

ГЛАВА 10

ВВОД-ВЫВОД

10А. ЧТЕНИЕ СТРОКИ С ТЕРМИНАЛА (RDLINE)

Считывается строка символов ASCII, заканчивающаяся возвратом каретки. При вводе управляющего символа УС + Н (08 шестнадцатеричное) удаляется последний введенный символ, а ввод УС + X (18 шестнадцатеричное) удаляет всю строку[1].

Если буфер переполняется, то на терминал посылается символ звукового сигнала (07 шестнадцатеричное). Выдается каждый символ, помещенный в буфер[2]. Для непечатаемых символов эхо выдается в виде знака вставки (∧), за которым следует печатаемый эквивалент (см. табл. 10.1). Перед выходом на терминал посылается последовательность символов для перехода на новую строку (обычно возврат каретки, перевод строки).

Таблица 10.1. *Управляющие символы* ASCII *и печатаемые эквиваленты*

Имя	Шестнадцатеричное значение	Печатаемый эквивалент	Имя	Шестнадцатеричное значение	Печатаемый эквивалент
NUL	00	УС + @	DLE	10	УС + P
SOH	01	УС + A	DC1	11	УС + Q
STX	02	УС + B	DC2	12	УС + R
ETX	03	УС + C	DC3	13	УС + S
EOT	04	УС + D	DC4	14	УС + T
ENQ	05	УС + E	NAK	15	УС + U
ACK	06	УС + F	SYN	16	УС + V
BEL	07	УС + G	ETB	17	УС + W

[1] Управляющие символы, вводимые при одновременном нажатии клавиш "Управляющий символ" (УС) и соответствующего символа, далее в тексте обозначают как УС + + символ. (*Прим. перев.*)

[2] Выдача на терминал введенного символа называют "эхом". (*Прим. перев.*)

Имя	Шестнадцатеричное значение	Печатаемый эквивалент	Имя	Шестнадцатеричное значение	Печатаемый эквивалент
BS	08	УС + H	CAN	18	УС + X
HT	09	УС + I	EM	19	УС + Y
LF	0A	УС + J	SUB	1A	УС + Z
VT	0B	УС + K	ESC	1B	УС + [
FF	0C	УС + L	FS	1C	УС + \
CR	0D	УС + M	GS	1D	УС +]
SO	0E	УС + N	RS	1E	УС + ∧
SI	0F	УС + O	VS	1F	УС + –

Подпрограмма RDLINE работает совместно со следующими системно-зависимыми подпрограммами:

1) RDCHAR читает символ с терминала и посылает его в аккумулятор;
2) WRCHAR посылает на терминал символ, находящийся в аккумуляторе;
3) WRNEWL выдает на терминал последовательность для новой строки.

Предполагается, что эти подпрограммы изменяют все регистры пользователя.

Подпрограмма RDLINE является примером программы, поддерживающей ввод с терминала. В реальной системе управляющие символы и подпрограммы ввода-вывода будут, конечно, зависеть от ЭВМ. Конкретный пример, приведенный в листинге, относится к ЭВМ, работающей под управлением операционной системы CP/M со стандартной базовой дисковой операционной системой (BDOS), доступ к которой выполняется при обращении к ячейке памяти 0005_{16}. В табл. 10.2 перечислены наиболее часто используемые функции BDOS CP/M. Для получения более полной информации по CP/M см. книгу Thom Hogan, Osborne CP/M User Guide, Second Edition (Berkeley: Osborne/McGraw-Hill, 1982).

Таблица 10.2. *Функции* BDOS *для CP/M 2.0*

Номер функции (десятичный в регистре C)	Название функции	Параметр на входе	Параметр на выходе
0	Сброс системы	Отсутствует	Отсутствует
1	Ввод с консоли	– " –	A = символ ASCII
2	Вывод на консоль	E = символ ASCII	Отсутствует
3	Ввод с устройства ввода перфоленты	Отсутствует	A = символ ASCII
4	Вывод на перфоратор	E = символ ASCII	Отсутствует
5	Вывод на устройство печати	E = символ ASCII	– " –
6	Прямой ввод с консоли	$E = FF_{16}$	A = символ ASCII или 00, если нет символа
6	Прямой вывод на консоль	E = символ ASCII	Отсутствует

Номер функции (десятичный в регистре С)	Название функции	Параметр на входе	Параметр на выходе
7	Получение байта состояния ввода-вывода	Отсутствует	A = IOBYTE
8	Установка байта состояния ввода-вывода	E = IOBYTE	Отсутствует
9	Выдача строки	DE = адрес строки	– " –
10	Считывание с консоли в буфер	DE = адрес буфера	(Данные в буфере)
11	Получение состояния консоли	Отсутствует	A = 00 (нет символа) или A = FF_{16} (символ готов)

Процедура. Цикл начинается с чтения символа. Если это символ возврата каретки, то на терминал посылается последовательность для перехода на новую строку и осуществляется выход из программы. В противном случае проверяется, не введен ли один из специальных символов УС + Н или УС + X. Если буфер не пустой, то УС + Н вызывает уменьшение указателя буфера и счетчика символов на 1 и выдачу на терминал последовательности для шага назад (курсор влево, пробел, курсор влево). При вводе УС + X символы удаляются до тех пор, пока не освободится буфер.

Если введен не специальный символ, то определяется, заполнен ли буфер. Если буфер заполнен, то на терминал посылается звуковой сигнал. Если нет, то запоминается символ в буфере, на терминал выдается эхо символа и увеличиваются счетчик символов и указатель буфера.

Перед выдачей эха символа или удалением символа с дисплея должно быть определено, является ли символ печатаемым. Если нет (т. е. это символ непечатаемого управляющего кода ASCII), то должны быть выданы или удалены два символа: указатель символа управления (стрелка вверх или знак вставки) и печатаемый эквивалент (см. табл. 10.1). Заметим, однако, что эти символы запоминаются в их непечатаемой форме.

Используемые регистры: все.
Время выполнения: приблизительно 158 (8080) или 154 (8085) тактов, чтобы поместить в буфер одинарный символ, без учета времени выполнения подпрограммы RDCHAR или WRCHAR.
Размер программы: 160 байт.
Память, необходимая для даных: отсутствует.
Специальные случаи:

1. Ввод УС + Н (удалить один символ) или УС + X (удалить всю строку) при пустом буфере не имеет эффекта.

2. Если буфер полон, обычный символ не принимается и на терминал выдается звуковой сигнал (звонок).

УСЛОВИЯ НА ВХОДЕ

Базовый адрес буфера в регистрах H и L.
Длина (размер) буфера в байтах в регистре A.

УСЛОВИЯ НА ВЫХОДЕ

Число символов в буфере в регистре A.

ПРИМЕРЫ

1. Данные: строка с клавиатуры содержит 'ENTERcr'.
   Результат: счетчик символов = 5 (длина строки), буфер содержит 'ENTER', на терминал посылается строка 'ENTER' и последовательность перехода на новую строку (обычно возврат каретки, перевод строки, или просто возврат каретки), заметим, что 'cr' (возврат каретки) в буфер не записывается.
2. Данные: строка с клавиатуры содержит 'DMУС+HNУС+XENTETУС+HRcr'.
   Результат: счетчик символов = 5 (длина окончательной строки), буфер содержит 'ENTER', на терминал посылается 'DMСтрокаВозвратаНаШагNСтрокаВозвратаНаШагСтрокаВозвратаНаШагENTERСтрокаВозвратаНаШагR', после чего идет последовательность перехода на новую строку. СтрокаВозвратаНаШаг удаляет один символ с экрана и перемещает курсор влево на одну позицию.

   Последовательность операций следующая:

Введенный символ	Буфер до ввода	Буфер после ввода	Посылается на терминал
D	Пустой	'D'	D
M	'D'	'DM'	M
УС + H	'DM'	'D'	Строка возврата на шаг
N	'D'	'DN'	N
УС + X	'DN'	Пустой	2 строки возврата на шаг
E	Пустой	'E'	E
N	'E'	'EN'	N
T	'EN'	'ENT'	T
E	'ENT'	'ENTE'	E
T	'ENTE'	'ENTET'	T
УС + H	'ENTET'	'ENTE'	Строка возврата на шаг
R	'ENTE'	'ENTER'	R
cr	'ENTER'	'ENTER'	Последовательность перехода на новую строку

Как происходил ввод:

а) оператор ввел 'D', 'M';

б) оператор увидел, что по ошибке ввел 'M' (должно быть 'N'), после чего ввел УС + + H, чтобы удалить 'M', а затем ввел 'N';

в) оператор увидел, что первый символ 'D' также введен неправильно (должно быть 'E'). Так как ошибка была не в последнем символе, оператор ввел УС + X, чтобы удалить всю строку, а затем ввел 'ENTET';

г) оператор увидел, что последнее 'T' введено неправильно (должно быть 'R'), ввел УС + H, чтобы удалить его и ввел 'R';

д) для окончания строки оператор ввел символ возврата каретки.

```
;                                                                  ;
;                                                                  ;
;                                                                  ;
;                                                                  ;
;       ЗАГОЛОВОК:      ЧТЕНИЕ СТРОКИ С ТЕРМИНАЛА
;       ИМЯ:            RDLINE                                     ;
;                                                                  ;
;                                                                  ;

;                                                                  ;
;       НАЗНАЧЕНИЕ:     ЧИТАЕТ СИМВОЛЫ С УСТРОЙСТВА CON: CP/M BDOS ДО  ;
;                       ТЕХ ПОР, ПОКА НЕ БУДЕТ ОБНАРУЖЕН СИМВОЛ ВОЗВРАТА;
;                       КАРЕТКИ. ВСЕ УПРАВЛЯЮЩИЕ СИМВОЛЫ, КРОМЕ        ;
;                       ПРИВЕДЕННЫХ НИЖЕ, ПОМЕЩАЮТСЯ В БУФЕР, ПРИ ЭТОМ ;
;                       ВЫВОДЯТСЯ ЭКВИВАЛЕНТНЫЕ ИМ ПЕЧАТАЕМЫЕ СИМВОЛЫ  ;
;                       ASCII С ПРЕДШЕСТВУЮЩЕЙ СТРЕЛКОЙ ВВЕРХ.         ;
;                         УПРАВЛЯЮЩИЙ СИМВОЛ H: УДАЛИТЬ ПОСЛЕДНИЙ СИМВОЛ;
;                         УПРАВЛЯЮЩИЙ СИМВОЛ X: УДАЛИТЬ ВСЮ СТРОКУ     ;
;                                                                  ;
;       ВХОД:           ПАРА РЕГИСТРОВ H = БАЗОВЫЙ АДРЕС БУФЕРА    ;
;                       РЕГИСТР A = ДЛИНА БУФЕРА В БАЙТАХ          ;
;                                                                  ;
;       ВЫХОД:          РЕГИСТР A = ЧИСЛО СИМВОЛОВ В БУФЕРЕ        ;
;                                                                  ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ:  ВСЕ                                ;
;                                                                  ;
;       ВРЕМЯ:          ЗАВИСИТ ОТ ДЛИТЕЛЬНОСТИ ВВОДА СТРОКИ       ;
;                                                                  ;
;       РАЗМЕР:         ПРОГРАММА - 160 БАЙТ                       ;
;                                                                  ;
;                                                                  ;

        ;ЭКВИВАЛЕНТНЫЕ ЗНАЧЕНИЯ
BELL    EQU     07H     ;СИМВОЛ ЗВОНКА (ВЫДАЕТ НА ТЕРМИНАЛ ЗВУКОВОЙ
                        ; СИГНАЛ)
BSKEY   EQU     08H     ;СИМВОЛ ВОЗВРАТА НА ШАГ НА КЛАВИАТУРЕ
CR      EQU     0DH     ;ВОЗВРАТ КАРЕТКИ НА КОНСОЛИ
CRKEY   EQU     0DH     ;СИМВОЛ ВОЗВРАТА КАРЕТКИ НА КЛАВИАТУРЕ
CSRLFT  EQU     08H     ;ПЕРЕМЕСТИТЬ КУРСОР НА КОНСОЛИ ВЛЕВО
DELKEY  EQU     18H     ;СИМВОЛ УДАЛЕНИЯ СТРОКИ НА КЛАВИАТУРЕ
LF      EQU     0AH     ;ПЕРЕВОД СТРОКИ ДЛЯ КОНСОЛИ
SPACE   EQU     20H     ;СИМВОЛ ПРОБЕЛА (ПОМЕЧАЕТ ТАКЖЕ КОНЕЦ УПРАВЛЯЮЩИХ
                        ; СИМВОЛОВ)
UPARRW  EQU     5EH     ;СТРЕЛКА ВВЕРХ, ИСПОЛЬЗУЕМАЯ ДЛЯ ИНДИКАЦИИ
                        ; УПРАВЛЯЮЩИХ СИМВОЛОВ

BDOS    EQU     0005H   ;ТОЧКА ВХОДА BDOS
DIRIO   EQU     6       ;ФУНКЦИЯ BDOS ПРЯМОГО ВВОДА-ВЫВОДА
PSTRG   EQU     9       ;ФУНКЦИЯ BDOS ПЕЧАТИ СТРОКИ
STERM   EQU     '$'     ;ТЕРМИНАТОР СТРОКИ В CP/M

RDLINE:
        MOV     C,A                     ;C = ДЛИНА БУФЕРА
                                        ;HL = УКАЗАТЕЛЬ БУФЕРА

        ;ЗАДАТЬ НАЧАЛЬНОЕ ЗНАЧЕНИЕ СЧЕТЧИКА СИМВОЛОВ РАВНЫМ НУЛЮ
INIT:
        MVI     B,0                     ;СЧЕТЧИК СИМВОЛОВ = 0
```

```
        ;ЧИТАТЬ СИМВОЛЫ, ПОКА НЕ БУДЕТ ВВЕДЕН СИМВОЛ ВОЗВРАТА КАРЕТКИ
RDLOOP:
        CALL    RDCHAR              ;ПРОЧИТАТЬ СИМВОЛ С КЛАВИАТУРЫ,
                                    ; НЕ ВЫДАВАЯ ЭХА

        ;ПРОВЕРИТЬ НА ВОЗВРАТ КАРЕТКИ, ЕСЛИ ВОЗВРАТ КАРЕТКИ - ВЫЙТИ
        CPI     CRKEY
        JZ      EXITRD              ;ЕСЛИ ВОЗВРАТ КАРЕТКИ, ТО КОНЕЦ СТРОКИ

        ;ПРОВЕРИТЬ НА СИМВОЛ ШАГА НАЗАД И, ЕСЛИ ШАГ НАЗАД, УДАЛИТЬ СИМВОЛ
        CPI     BSKEY
        JNZ     RDLP1               ;ПЕРЕЙТИ, ЕСЛИ НЕ ШАГ НАЗАД
        CALL    BACKSP              ;ЕСЛИ ШАГ НАЗАД, ТО УДАЛИТЬ ОДИН СИМВОЛ,
        JMP     RDLOOP              ; ЗАТЕМ СНОВА НАЧАТЬ ЦИКЛ ЧТЕНИЯ

        ;ПРОВЕРИТЬ НА СИМВОЛ УДАЛЕНИЯ СТРОКИ И, ЕСЛИ УДАЛЕНИЕ СТРОКИ,
        ; ОСВОБОДИТЬ БУФЕР
RDLP1:
        CPI     DELKEY
        JNZ     RDLP2               ;ПЕРЕЙТИ, ЕСЛИ НЕ СИМВОЛ УДАЛЕНИЯ СТРОКИ
DEL1:
        CALL    BACKSP              ;УДАЛИТЬ СИМВОЛ
        JNZ     DEL1                ;ПРОДОЛЖАТЬ, ПОКА БУФЕР НЕ ОСВОБОДИТСЯ
                                    ;В ДЕЙСТВИТЕЛЬНОСТИ ЭТО ВОЗВРАТ НА ШАГ
                                    ; ДЛЯ КАЖДОГО СИМВОЛА, А НЕ ПЕРЕХОД
                                    ; НА СТРОКУ ВВЕРХ
        JMP     RDLOOP

        ;НЕ СПЕЦИАЛЬНЫЙ СИМВОЛ
        ; ПРОВЕРИТЬ НЕ ЗАПОЛНЕНА ЛИ СТРОКА
        ; ЕСЛИ ЗАПОЛНЕНА, ТО ВЫДАТЬ ЗВУКОВОЙ СИГНАЛ И ПРОДОЛЖАТЬ
        ; ЕСЛИ НЕ ЗАПОЛНЕНА,ТО ЗАПОМНИТЬ СИМВОЛ И ВЫДАТЬ НА КОНСОЛЬ
RDLP2:
        MOV     E,A                 ;СОХРАНИТЬ СИМВОЛ
        MOV     A,B                 ;БУФЕР ПОЛОН?
        CMP     C                   ; СРАВНИТЬ СЧЕТЧИК И ДЛИНУ БУФЕРА
        JC      STRCH               ;ПЕРЕЙТИ, ЕСЛИ БУФЕР НЕ ЗАПОЛНЕН
        MVI     A,BELL              ;ЗАПОЛНЕН, ВЫДАТЬ НА ТЕРМИНАЛ ЗВУКОВОЙ
        CALL    WRCHAR              ; СИГНАЛ
        JMP     RDLOOP              ;ЗАТЕМ ПРОДОЛЖАТЬ ЦИКЛ ЧТЕНИЯ

        ;БУФЕР НЕ ЗАПОЛНЕН, ЗАПОМНИТЬ СИМВОЛ
STRCH:
        MOV     A,E                 ;ВЗЯТЬ СИМВОЛ НАЗАД
        MOV     M,A                 ;ЗАПОМНИТЬ СИМВОЛ В БУФЕРЕ
        INX     H                   ;УВЕЛИЧИТЬ УКАЗАТЕЛЬ БУФЕРА
        INR     B                   ;УВЕЛИЧИТЬ СЧЕТЧИК СИМВОЛОВ

        ;ЕСЛИ ЭТО УПРАВЛЯЮЩИЙ СИМВОЛ, ТО ВЫВЕСТИ
        ; СТРЕЛКУ ВВЕРХ И ВЫДАВАЕМЫЙ НА ПЕЧАТЬ ЭКВИВАЛЕНТ
        CPI     SPACE               ;ЕСЛИ МЕНЬШЕ, ЧЕМ ПРОБЕЛ
                                    ; (ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО 20),
                                    ; ТО УПРАВЛЯЮЩИЙ СИМВОЛ
        JNC     PRCH                ;ПЕРЕЙТИ, ЕСЛИ ПЕЧАТАЕМЫЙ СИМВОЛ
        PUSH    PSW                 ;СОХРАНИТЬ СИМВОЛ
        MVI     A,UPARRW            ;ВЫДАТЬ СТРЕЛКУ ВВЕРХ
        CALL    WRCHAR
```

```
        POP     PSW             ;ВОССТАНОВИТЬ СИМВОЛ
        ADI     40H             ;ИЗМЕНИТЬ НА ПЕЧАТАЕМУЮ ФОРМУ
PRCH:   CALL    WRCHAR          ;ВЫДАТЬ СИМВОЛ НА ТЕРМИНАЛ
        JMP     RDLOOP          ;ЗАТЕМ ПРОДОЛЖИТЬ ЦИКЛ ЧТЕНИЯ

        ;ВЫХОД
        ;ПОСЛАТЬ НА ТЕРМИНАЛ ПОСЛЕДОВАТЕЛЬНОСТЬ СИМВОЛОВ ДЛЯ ПЕРЕХОДА
        ; НА НОВУЮ СТРОКУ (ОБЫЧНО ВОЗВРАТ КАРЕТКИ, ПЕРЕВОД СТРОКИ)
        ;ДЛИНА СТРОКИ = СЧЕТЧИКУ СИМВОЛОВ
EXITRD:
        CALL    WRNEWL          ;ВЫДАТЬ ПОСЛЕДОВАТЕЛЬНОСТЬ ПЕРЕХОДА
                                ; НА НОВУЮ СТРОКУ
        MOV     A,B             ;ДЛИНА СТРОКИ = СЧЕТЧИКУ СИМВОЛОВ
        RET

        ;****************************************
        ;ПОДПРОГРАММА: RDCHAR
        ;НАЗНАЧЕНИЕ:   ЧИТАЕТ СИМВОЛ БЕЗ ВЫДАЧИ ЕГО НА УСТРОЙСТВО ВЫВОДА
        ;ВХОД:  НЕТ ПАРАМЕТРОВ
        ;ВЫХОД: РЕГИСТР A = СИМВОЛ
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF,DE
        ;****************************************
RDCHAR:
        PUSH    H               ;СОХРАНИТЬ HL,BC
        PUSH    B

        ;ЖДАТЬ СИМВОЛ С КОНСОЛИ
RDWAIT:
        MVI     C,DIRIO         ;ПРЯМОЙ ВВОД-ВЫВОД НА КОНСОЛЬ
        MVI     E,0FFH          ;ЗАДАТЬ ВВОД
        CALL    BDOS            ;ПРОЧИТАТЬ СИМВОЛ С КОНСОЛИ
        ORA     A               ;ЗАЦИКЛИТЬ, ЕСЛИ НЕТ СИМВОЛА (A = 0)
        JZ      RDWAIT

        POP     B               ;ВОССТАНОВИТЬ BC,HL
        POP     H
        RET                     ;ВОЗВРАТИТЬСЯ С СИМВОЛОМ В РЕГИСТРЕ A

        ;****************************************
        ;ПОДПРОГРАММА: WRCHAR
        ;НАЗНАЧЕНИЕ:   ЗАПИСЫВАЕТ СИМВОЛ НА УСТРОЙСТВО ВЫВОДА
        ;ВХОД:  РЕГИСТР A = СИМВОЛ
        ;ВЫХОД: НЕТ ПАРАМЕТРОВ
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF,DE
        ;****************************************
WRCHAR:
        PUSH    H               ;СОХРАНИТЬ HL,BC
        PUSH    B

        ;ЗАПИСАТЬ СИМВОЛ
        MVI     C,DIRIO         ;ПРЯМОЙ ВВОД-ВЫВОД НА КОНСОЛЬ
        MOV     E,A             ;ЗАДАТЬ ВЫВОД - СИМВОЛ В E
        CALL    BDOS            ;ЗАПИСАТЬ СИМВОЛ НА КОНСОЛЬ
        POP     B               ;ВОССТАНОВИТЬ BC,HL
        POP     H
        RET
```

```
        ;******************************************
        ;ПОДПРОГРАММА: WRNEWL
        ;НАЗНАЧЕНИЕ:   ВЫДАЕТ НА ТЕРМИНАЛ ПОСЛЕДОВАТЕЛЬНОСТЬ СИМВОЛОВ
        ;              ДЛЯ ПЕРЕХОДА НА НОВУЮ СТРОКУ. ОБЫЧНО ЭТА
        ;              ПОСЛЕДОВАТЕЛЬНОСТЬ ВКЛЮЧАЕТ СИМВОЛЫ ВОЗВРАТА
        ;              КАРЕТКИ И ПЕРЕВОДА СТРОКИ. НА НЕКОТОРЫХ ЭВМ
        ;              ТРЕБУЕТСЯ ТОЛЬКО ВОЗВРАТ КАРЕТКИ.
        ;ВХОД:  НЕТ ПАРАМЕТРОВ
        ;ВЫХОД: НЕТ ПАРАМЕТРОВ
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF,DE
        ;******************************************
WRNEWL:
        PUSH    H                   ;СОХРАНИТЬ HL,BC
        PUSH    B

        ;ПОСЛАТЬ НА ТЕРМИНАЛ ПОСЛЕДОВАТЕЛЬНОСТЬ СИМВОЛОВ ДЛЯ
        ; ПЕРЕХОДА НА НОВУЮ СТРОКУ
        LXI     D,NLSTRG            ;ЗАДАТЬ АДРЕС ПОСЛЕДОВАТУЕЛЬНОСТИ
                                    ; СИМВОЛОВ  ДЛЯ ПЕРЕХОДА НА НОВУЮ
                                    ; СТРОКУ
        CALL    WRSTRG              ;ПОСЛАТЬ СТРОКУ НА ТЕРМИНАЛ
        POP     B                   ;ВОССТАНОВИТЬ BC,HL
        POP     H
        RET

NLSTRG: DB      CR,LF,STERM         ;СТРОКА ДЛЯ ПЕРЕХОДА НА НОВУЮ СТРОКУ
                                    ; ЗАМЕТИМ, ЧТО $ (STERM) ЯВЛЯЕТСЯ
                                    ; ТЕРМИНАТОРОМ В CP/M

        ;******************************************
        ;ПОДПРОГРАММА: BACKSP
        ;НАЗНАЧЕНИЕ:   ВЫПОЛНЯЕТ ШАГ НАЗАД
        ;ВХОД:  B = ЧИСЛО СИМВОЛОВ В БУФЕРЕ
        ;       HL = СЛЕДУЮЩИЙ ДОСТУПНЫЙ АДРЕС В БУФЕРЕ
        ;ВЫХОД: ЕСЛИ В БУФЕРЕ НЕТ СИМВОЛОВ, ТО
        ;         Z = 1
        ;       ИНАЧЕ
        ;         Z = 0
        ;       СИМВОЛ УДАЛЯЕТСЯ ИЗ БУФЕРА
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF,B,DE
        ;******************************************
BACKSP:
        ;ПРОВЕРИТЬ, ПУСТОЙ ЛИ БУФЕР
        MOV     A,B                 ;ПРОВЕРИТЬ ЧИСЛО СИМВОЛОВ
        ORA     A
        RZ                          ;ВЫЙТИ, ЕСЛИ БУФЕР ПУСТОЙ

        ;ВЫВЕСТИ СТРОКУ ДЛЯ ШАГА НАЗАД, ЧТОБЫ УДАЛИТЬ СИМВОЛ С ДИСПЛЕЯ
        DCX     H                   ;УМЕНЬШИТЬ УКАЗАТЕЛЬ БУФЕРА
        PUSH    H                   ;СОХРАНИТЬ HL,BC
        PUSH    B
        MOV     A,M                 ;ВЗЯТЬ СИМВОЛ
        CPI     20H                 ;ЭТО УПРАВЛЯЮЩИЙ СИМВОЛ?
        JNC     BS1                 ; НЕТ, ПЕРЕЙТИ, УДАЛИТЬ ТОЛЬКО ОДИН
                                    ; СИМВОЛ
        LXI     D,BSSTRG            ; ДА, УДАЛИТЬ ДВА СИМВОЛА
                                    ; (СТРЕЛКУ ВВЕРХ И ПЕЧАТАЕМЫЙ ЭКВИВАЛЕНТ)
```

```
        CALL    WRSTRG              ;ЗАПИСАТЬ ПОСЛЕДОВАТЕЛЬНОСТЬ СИМВОЛОВ
                                    ; ДЛЯ ШАГА НАЗАД
BS1:    LXI     D,BSSTRG
        CALL    WRSTRG              ;ЗАПИСАТЬ ПОСЛЕДОВАТЕЛЬНОСТЬ СИМВОЛОВ
                                    ; ДЛЯ ШАГА НАЗАД
        POP     B                   ;ВОССТАНОВИТЬ BC,HL
        POP     H

        ;УМЕНЬШИТЬ СЧЕТЧИК СИМВОЛОВ НА 1
        DCR     B                   ;УМЕНЬШИТЬ БУФЕР НА ОДИН СИМВОЛ
        RET

        ;СТРОКА ШАГА НАЗАД ДЛЯ ТЕРМИНАЛА
        ;ПЕРЕМЕЩАЕТ КУРСОР ВЛЕВО, ВЫДАЕТ ПРОБЕЛ НА ТЕРМИНАЛ,
        ; ПЕРЕМЕЩАЕТ КУРСОР ВЛЕВО
        ;ЗАМЕТИМ: STERM (¤) ЯВЛЯЕТСЯ ТЕРМИНАТОРОМ В CP/M
BSSTRG: DB      CSRLFT,SPACE,CSRLFT,STERM

        ;****************************************
        ;ПОДПРОГРАММА: WRSTRG
        ;НАЗНАЧЕНИЕ:   ВЫВОДИТ СТРОКУ НА КОНСОЛЬ
        ;ВХОД:  HL = БАЗОВЫЙ АДРЕС СТРОКИ
        ;ВЫХОД: НЕТ ПАРАМЕТРОВ
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF,DE,HL
        ;****************************************
WRSTRG:
        PUSH    B                   ;СОХРАНИТЬ BC
        MVI     C,PSTRG             ;ФУНКЦИЯ - ПЕЧАТЬ СТРОКИ
        CALL    BDOS                ;ВЫВЕСТИ СТРОКУ, ЗАКАНЧИВАЮЩУЮСЯ
                                    ; СИМВОЛОМ ¤
        POP     B                   ;ВОССТАНОВИТЬ BC
        RET


;                                                                   ;
;                                                                   ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                           ;
;                                                                   ;
;                                                                   ;
;                                                                   ;

        ;ЭКВИВАЛЕНТНЫЕ ЗНАЧЕНИЯ
PROMPT  EQU     '?'                 ;ПОДСКАЗКА ОПЕРАТОРУ - ЗНАК ВОПРОСА

SC10A:
        ;ПРОЧИТАТЬ СТРОКУ С ТЕРМИНАЛА
        MVI     A,PROMPT            ;ВЫВЕСТИ ПОДСКАЗКУ (?)
        CALL    WRCHAR
        LXI     H,INBUFF            ;HL = АДРЕС БУФЕРА ВВОДА
        MVI     A,LINBUF            ;A = ДЛИНА БУФЕРА
        CALL    RDLINE              ;ПРОЧИТАТЬ СТРОКУ
        ORA     A                   ;ПРОВЕРИТЬ ДЛИНУ СТРОКИ
        JZ      SC10A               ;ЕСЛИ ДЛИНА СТРОКИ 0, ВВЕСТИ СЛЕДУЮЩУЮ
                                    ; СТРОКУ

        ;ВЫДАТЬ СТРОКУ НА КОНСОЛЬ
        MOV     B,A                 ;СОХРАНИТЬ ЧИСЛО СИМВОЛОВ В БУФЕРЕ
        LXI     H,INBUFF            ;УСТАНОВИТЬ УКАЗАТЕЛЬ НА НАЧАЛО БУФЕРА
```

```
TLOOP:
        MOV     A,M             ;ВЫВЕСТИ СЛЕДУЮЩИЙ СИМВОЛ
        CALL    WRCHAR
        INX     H               ;УВЕЛИЧИТЬ УКАЗАТЕЛЬ БУФЕРА
        DCR     B               ;УМЕНЬШИТЬ СЧЕТЧИК СИМВОЛОВ
        JNZ     TLOOP           ;ПРОДОЛЖАТЬ, ПОКА НЕ БУДУТ ПОСЛАНЫ
                                ; ВСЕ СИМВОЛЫ
        CALL    WRNEWL          ;ЗАТЕМ ЗАКОНЧИТЬ ВОЗВРАТОМ КАРЕТКИ И
                                ; ПЕРЕВОДОМ СТРОКИ

        JMP     SC10A

        ;СЕКЦИЯ ДАННЫХ
LINBUF  EQU     16              ;ДЛИНА БУФЕРА ВВОДА
INBUFF: DS      LINBUF          ;БУФЕР ВВОДА

        END
```

10B. ЗАПИСЬ СТРОКИ НА УСТРОЙСТВО ВЫВОДА (WRLINE)

Записываются символы до освобождения буфера, заданного длиной и базовым адресом. Работа осуществляется совместно с системно-зависимой подпрограммой WRCHAR, которая посылает символ, находящийся в аккумуляторе, на устройство вывода.

Подпрограмма WRLINE является примером драйвера вывода. Действительные подпрограммы ввода-вывода будут, конечно, зависеть от ЭВМ. Конкретный пример, приведенный в листинге, относится к ЭВМ, работающей под управлением операционной системы CP/M со стандартной базовой дисковой операционной системой (BDOS), доступ к которой выполняется при обращении к ячейке памяти с адресом 0005_{16}.

Процедура. Если буфер пустой, то осуществляется немедленный выход из программы. В противном случае на устройство вывода посылается по одному символу до освобождения буфера. Чтобы избежать зависимости от WRCHAR, все временные данные сохраняются в памяти, а не в регистрах.

Используемые регистры: AF, BC, DE, HL.
Время выполнения: 20 тактов плюс 44 такта на байт (8080) или 16 тактов плюс 45 тактов на байт (8085), без учета времени выполнения подпрограммы.
Размер программы: 24 байта.
Память, необходимая для данных: отсутствует.
Специальный случай: пустой буфер вызывает немедленный выход, при этом на устройство вывода ничего не выдается.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес буфера в регистрах H и L.
Число символов в буфере в регистре A.

УСЛОВИЯ НА ВЫХОДЕ

Нет параметров

ПРИМЕР

1. Данные: число символов = 5,
   буфер содержит 'ENTER'.
   Результат: на устройство вывода выдается 'ENTER'.

```
;                                                                    ;
;                                                                    ;
;                                                                    ;
;                                                                    ;
;       ЗАГОЛОВОК:      ЗАПИСЬ СТРОКИ НА УСТРОЙСТВО ВЫВОДА           ;
;       ИМЯ:            WRLINE                                       ;
;                                                                    ;
;                                                                    ;

;                                                                    ;
;       НАЗНАЧЕНИЕ:     ЗАПИСЫВАЕТ СИМВОЛЫ НА УСТРОЙСТВО CON: CP/M BDOS ;
;                                                                    ;
;       ВХОД:           ПАРА РЕГИСТРОВ H = БАЗОВЫЙ АДРЕС БУФЕРА      ;
;                       РЕГИСТР A = ЧИСЛО СИМВОЛОВ В БУФЕРЕ          ;
;                                                                    ;
;       ВЫХОД:          НЕТ ПАРАМЕТРОВ                               ;
;                                                                    ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ:  ВСЕ                                  ;
;                                                                    ;
;       ВРЕМЯ:          ЗАВИСИТ ОТ ДЛИНЫ СТРОКИ                      ;
;                                                                    ;
;       РАЗМЕР:         ПРОГРАММА - 24 БАЙТА                         ;
;                                                                    ;
;                                                                    ;

        ;ЭКВИВАЛЕНТНЫЕ ЗНАЧЕНИЯ
BDOS    EQU     0005H   ;ТОЧКА ВХОДА BDOS
DIRIO   EQU     6       ;ФУНКЦИЯ BDOS ПРЯМОГО ВВОДА-ВЫВОДА

WRLINE:
        ;ЕСЛИ БУФЕР ПУСТОЙ, НЕМЕДЛЕННО ВЫЙТИ
        ORA     A               ;ПРОВЕРИТЬ ЧИСЛО СИМВОЛОВ
        RZ                      ;ВЫЙТИ, ЕСЛИ БУФЕР ПУСТОЙ
        MOV     B,A             ;B = СЧЕТЧИК
                                ;HL = БАЗОВЫЙ АДРЕС БУФЕРА

        ;ЦИКЛ ПЕРЕДАЧИ СИМВОЛОВ НА УСТРОЙСТВО ВЫВОДА
WRLLP:
        MOV     A,M             ;ВЗЯТЬ СЛЕДУЮЩИЙ СИМВОЛ
        CALL    WRCHAR          ;ПОСЛАТЬ СИМВОЛ
        INX     H               ;УВЕЛИЧИТЬ УКАЗАТЕЛЬ БУФЕРА
        DCR     B               ;УМЕНЬШИТЬ СЧЕТЧИК
        JNZ     WRLLP           ;ПРОДОЛЖАТЬ, ПОКА НЕ БУДУТ ПОСЛАНЫ
                                ; ВСЕ СИМВОЛЫ

        RET                     ;ВЫЙТИ
        ;****************************************
        ;ПОДПРОГРАММА: WRCHAR
        ;НАЗНАЧЕНИЕ:   ЗАПИСЫВАЕТ СИМВОЛ НА УСТРОЙСТВО ВЫВОДА
        ;ВХОД:  РЕГИСТР A = СИМВОЛ
        ;ВЫХОД: НЕТ ПАРАМЕТРОВ
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF,DE
        ;****************************************
WRCHAR:
        PUSH    H               ;СОХРАНИТЬ HL,BC
        PUSH    B
        MVI     C,DIRIO         ;ПРЯМОЙ ВВОД-ВЫВОД НА КОНСОЛЬ
        MOV     E,A             ;СИМВОЛ В РЕГИСТРЕ E
```

```
        CALL    BDOS                ;ВЫДАТЬ СИМВОЛ

        POP     B                   ;ВОССТАНОВИТЬ BC,HL
        POP     H
        RET
```

ПРИМЕР ВЫПОЛНЕНИЯ

```
RCBUF   EQU     10                  ;ФУНКЦИЯ BDOS ЧТЕНИЯ С КОНСОЛИ В БУФЕР

        ;ФУНКЦИЯ BDOS ЧТЕНИЯ С КОНСОЛИ В БУФЕР ИСПОЛЬЗУЕТ СЛЕДУЮЩИЙ
        ; ФОРМАТ БУФЕРА:
        ;  БАЙТ 0: ДЛИНА БУФЕРА (МАКСИМАЛЬНОЕ ЧИСЛО СИМВОЛОВ)
        ;  БАЙТ 1: ЧИСЛО ПРОЧИТАННЫХ СИМВОЛОВ (ДЛИНА СТРОКИ)
        ;  БАЙТ 2 И ДАЛЕЕ: ВВЕДЕННЫЕ СИМВОЛЫ

        ;ЭКВИВАЛЕНТНЫЕ СИМВОЛЫ
CR      EQU     0DH                 ;ВОЗВРАТ КАРЕТКИ ДЛЯ КОНСОЛИ
LF      EQU     0AH                 ;ПЕРЕВОД СТРОКИ ДЛЯ КОНСОЛИ
PROMPT  EQU     '?'                 ;ПОДСКАЗКА ОПЕРАТОРУ - ЗНАК ВОПРОСА

SC10B:
        ;ПРОЧИТАТЬ СТРОКУ С КОНСОЛИ
        MVI     A,PROMPT            ;ВЫВЕСТИ ПОДСКАЗКУ (?)
        CALL    WRCHAR
        LXI     B,INBUFF            ;ЗАДАТЬ АДРЕС БУФЕРА ВВОДА
        MVI     C,RCBUF             ;ФУНКЦИЯ BDOS ЧТЕНИЯ СТРОКИ
        CALL    BDOS                ;ПРОЧИТАТЬ СТРОКУ С КОНСОЛИ
        MVI     A,LF                ;ВЫДАТЬ СИМВОЛ ПЕРЕВОДА СТРОКИ
        CALL    WRCHAR

        ;ВЫДАТЬ СТРОКУ НА КОНСОЛЬ
        LXI     H,INBUFF+1          ;ЗАДАТЬ АДРЕС ЧИСЛА СИМВОЛОВ
                                    ; В БУФЕРЕ
        MOV     A,M                 ;ВЗЯТЬ ЧИСЛО СИМВОЛОВ
        INX     H                   ;ПОЛУЧИТЬ АДРЕС ПЕРВОГО БАЙТА ДАННЫХ
        CALL    WRLINE              ;ЗАПИСАТЬ СТРОКУ
        LXI     H,CRLF              ;ВЫВЕСТИ СИМВОЛЫ ВОЗВРАТА КАРЕТКИ,
                                    ; ПЕРЕВОДА СТРОКИ
        MVI     A,2                 ;ДЛИНА СТРОКИ CRLF
        CALL    WRLINE              ;ВЫВЕСТИ СТРОКУ CRLF

        JMP     SC10B               ;ПРОДОЛЖАТЬ

        ;СЕКЦИЯ ДАННЫХ
CRLF:   DB      CR,LF               ;ВОЗВРАТ КАРЕТКИ, ПЕРЕВОД СТРОКИ
LINBUF  EQU     10H                 ;ДЛИНА БУФЕРА ВВОДА
INBUFF: DB      LINBUF              ;ДЛИНА БУФЕРА ВВОДА
        DS      LINBUF              ;БУФЕР ДАННЫХ

        END
```

10C. ПРОВЕРКА И ГЕНЕРАЦИЯ 16-РАЗРЯДНОГО КОДА КОНТРОЛЯ ПО ИЗБЫТОЧНОСТИ (ICRC16, CRC16, GCRC16)

Генерируется 16-разрядный код циклической проверки по избыточности (CRC), основанный на протоколе IBM двоичной синхронной связи (BSC или Bisync). Используется полином $X^{16} + X^{15} + X^2 + 1$. Подпрограмма ICRC16 устанавливает CRC в 0 и записывает в полином соответствующий набор разрядов. Подпрограмма CRC16 объединяет предыдущий код CRC с CRC, сгенерированным для текущего байта данных. Подпрограмма GCRC16 возвращает CRC.

Процедура. Подпрограмма ICRC16 инициализирует CRC в 0 и записывает 1 в каждый разряд полинома, соответствующий степени X в формуле. Подпрограмма CRC16 обновляет CRC для байта данных. Она сдвигает как данные, так и CRC влево восемь раз; после каждого сдвига выполняется операция ИСКЛЮЧАЮЩЕЕ ИЛИ CRC и полинома, если операция ИСКЛЮЧАЮЩЕЕ ИЛИ разряда данных и старшего по значению разряда CRC дает в результате 1. Подпрограмма CRC16 оставляет CRC в ячейках памяти CRC (младший по значению байт) и CRC + 1 (старший по значению байт). Подпрограмма GCRC16 загружает значение CRC в регистры H и L.

Используемые регистры:
1. ICRC16: HL.
2. CRC16: отсутствуют.
3. GCRC16: HL.

Время выполнения:
1. ICRC16: 62 такта (8080 или 8085).
2. CRC16: 153 (8080) или 154 (8085) такта плюс в среднем 632 (8080) или 568 (8085) тактов на байт данных при условии, что для предыдущего кода CRC и полинома операция ИСКЛЮЧАЮЩЕЕ ИЛИ должна выполняться в половине итераций.
3. GCRC16: 26 тактов (8080 или 8085).

Размер программы:
1. ICRC16: 13 байт.
2. CRC16: 42 байта.
3. GCRC16: 4 байта.

Память, необходимая для данных: 4 байта в любом месте ОЗУ для CRC (2 байта, начинающиеся с адреса CRC) и полинома (2 байта, начинающиеся с адреса PLY).

УСЛОВИЯ НА ВХОДЕ

1. ICRC16: нет параметров.
2. CRC16: байт данных в регистре A, предыдущий код CRC в ячейках памяти CRC (младший по значению байт) и CRC+1 (старший по значению байт), полином CRC в ячейках памяти PLY (младший по значению байт) и PLY+1 (старший по значению байт).
3. GCRC16: код CRC в ячейках памяти CRC (младший по значению байт) и CRC+1 (старший по значению байт).

УСЛОВИЯ НА ВЫХОДЕ

1. ICRC16: 0 (начальное значение CRC) в ячейках памяти CRC (младший по значению байт) и CRC+1 (старший по значению байт), полином CRC в

ячейках памяти PLY (младший по значению байт) и PLY+1 (старший по значению байт).

2. CRC16: код CRC с учетом текущего байта данных в ячейках памяти CRC (младший по значению байт) и CRC+1 (старший по значению байт).

3. GCRC16: код CRC в регистрах H и L.

ПРИМЕРЫ

1. Генерация CRC:
   а) вызвать ICRC16 для начальной установки CRC в 0;
   б) вызывать CRC16 для обновления кода CRC по байту данных;
   в) вызвать GCRC16 для получения окончательного значения кода CRC.
2. Проверка CRC:
   а) вызвать ICRC16 для начальной установки CRC в 0;
   б) с целью проверки вызывать CRC16 для обновления CRC по каждому байту данных (включая сохраненное значение CRC);
   в) вызвать GCRC16 для получения окончательного значения кода CRC; если не было ошибок, оно должно быть равно 0.

Заметим, что только ICRC16 зависит от конкретного используемого полинома CRC. Чтобы изменить полином, нужно просто изменить данные ICRC16, загружаемые в ячейки памяти PLY (младший по значению байт) и PLY+1 (старший по значению байт).

```
;                                                                       ;
;                                                                       ;
;                                                                       ;
;                                                                       ;
;       ЗАГОЛОВОК:      ПРОВЕРКА И ГЕНЕРАЦИЯ 16-РАЗРЯДНОГО КОДА         ;
;                       КОНТРОЛЯ ПО ИЗБЫТОЧНОСТИ (CRC-16)               ;
;       ИМЯ:            ICRC16, CRC16, GCRC16                           ;
;                                                                       ;
;                                                                       ;

;                                                                       ;
;       НАЗНАЧЕНИЕ:                                                     ;
;                       ПРОТОКОЛ СИНХРОННОЙ СВЯЗИ. CRC БАЗИРУЕТСЯ       ;
;                       НА СЛЕДУЮЩЕМ ПОЛИНОМЕ:                          ;
;                         (^ ОЗНАЧАЕТ "ВОЗВЕСТИ В СТЕПЕНЬ")             ;
;                               X^16 + X^15 + X^2 + 1                   ;
;                                                                       ;
;                       ЧТОБЫ СГЕНЕРИРОВАТЬ CRC:                        ;
;                           1) ВЫЗВАТЬ ICRC16 ДЛЯ ИНИЦИАЛИЗАЦИИ         ;
;                              ПОЛИНОМА CRC И ОЧИСТКИ CRC.              ;
;                           2) ВЫЗВАТЬ CRC16 ДЛЯ КАЖДОГО БАЙТА ДАННЫХ.  ;
;                           3) ВЫЗВАТЬ GCRC16 ДЛЯ ПОЛУЧЕНИЯ CRC.        ;
;                              ЗАТЕМ CRC НЕОБХОДИМО ДОБАВИТЬ К ДАННЫМ,  ;
;                              ПРИЧЕМ СТАРШИЙ БАЙТ ДОЛЖЕН БЫТЬ ПЕРВЫМ.  ;
;                                                                       ;
;                       ЧТОБЫ ПРОВЕРИТЬ CRC:                            ;
;                           1) ВЫЗВАТЬ ICRC16 ДЛЯ ИНИЦИАЛИЗАЦИИ CRC.    ;
;                           2) ВЫЗВАТЬ CRC16 ДЛЯ КАЖДОГО БАЙТА ДАННЫХ И ;
;                              ДВУХ БАЙТ CRC, СГЕНЕРИРОВАННОГО РАНЕЕ.   ;
;                           3) ВЫЗВАТЬ GCRC16 ДЛЯ ПОЛУЧЕНИЯ CRC. ЕСЛИ   ;
;                              НЕТ ОШИБОК, ТО ЗНАЧЕНИЕ CRC ДОЛЖНО БЫТЬ  ;
;                              РАВНО НУЛЮ.                              ;
;                                                                       ;
```

```
;       ВХОД:              ICRC16 - НЕТ ПАРАМЕТРОВ                          ;
;                          CRC16  - РЕГИСТРА = БАЙТ ДАННЫХ                  ;
;                          GCRC16 - НЕТ ПАРАМЕТРОВ                          ;
;                                                                           ;
;       ВЫХОД:             ICRC16 - ИНИЦИАЛИЗИРУЮТСЯ CRC И PLY              ;
;                          CRC16  - ОБНОВЛЯЕТСЯ CRC                         ;
;                          GCRC16 - H И L = CRC                             ;
;                                                                           ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ:  ICRC16 - HL                                 ;
;                               CRC16  - НЕТ ПАРАМЕТРОВ                     ;
;                               GCRC16 - HL                                 ;
;                                                                           ;
;       ВРЕМЯ:             153 ТАКТА ПЛЮС 632 ТАКТА НА БАЙТ ДАННЫХ          ;
;                          ДЛЯ 8080                                         ;
;                          154 ТАКТА ПЛЮС 568 ТАКТОВ НА БАЙТ ДАННЫХ         ;
;                          ДЛЯ 8085                                         ;
;                          СЧИТАЕТСЯ, ЧТО ПОЛОВИНА ИТЕРАЦИЙ ТРЕБУЕТ         ;
;                          ВЫПОЛНЕНИЯ ОПЕРАЦИИ ИСКЛЮЧАЮЩЕЕ ИЛИ CRC И        ;
;                          ПОЛИНОМА                                         ;
;                                                                           ;
;       РАЗМЕР:            ПРОГРАММА - 59 БАЙТ                              ;
;                          ДАННЫЕ    - 4 БАЙТА                              ;
;                                                                           ;
;                                                                           ;

CRC16:
        ;СОХРАНИТЬ ВСЕ РЕГИСТРЫ
        PUSH    PSW
        PUSH    B
        PUSH    D
        PUSH    H

        ;ЦИКЛ ДЛЯ КАЖДОГО РАЗРЯДА ДАННЫХ, ГЕНЕРИРУЮЩИЙ CRC
        MVI     B,8             ;8 РАЗРЯДОВ НА БАЙТ
        LHLD    PLY             ;ПЕРЕСЛАТЬ ПОЛИНОМ В DE
        XCHG
        LHLD    CRC             ;ВЗЯТЬ ТЕКУЩЕЕ ЗНАЧЕНИЕ CRC
CRCLP:
        MOV     C,A             ;СОХРАНИТЬ ДАННЫЕ
        ANI     10000000B       ;ВЗЯТЬ РАЗРЯД 7 ДАННЫХ
        XRA     H               ;ВЫПОЛНИТЬ ОПЕРАЦИЮ ИСКЛЮЧАЮЩЕЕ ИЛИ
                                ; РАЗРЯДА 7 С РАЗРЯДОМ 15 CRC
        MOV     H,A
        DAD     H               ;СДВИНУТЬ CRC ВЛЕВО
        JNC     CRCLP1          ;ПЕРЕЙТИ, ЕСЛИ РАЗРЯД 7 ПРИ ОПЕРАЦИИ
                                ; ИСКЛЮЧАЮЩЕЕ ИЛИ БЫЛ РАВЕН 0

        ;РАЗРЯД 7 БЫЛ РАВЕН 1, ПОЭТОМУ ВЫПОЛНИТЬ ОПЕРАЦИЮ ИСКЛЮЧАЮЩЕЕ
        ; ИЛИ CRC С ПОЛИНОМОМ
        MOV     A,E             ;ВЗЯТЬ МЛАДШИЙ БАЙТ ПОЛИНОМА
        XRA     L               ;ВЫПОЛНИТЬ ОПЕРАЦИЮ ИСКЛЮЧАЮЩЕЕ ИЛИ
                                ; С НИЖНИМ БАЙТОМ CRC
        MOV     L,A
        MOV     A,D             ;ВЗЯТЬ СТАРШИЙ БАЙТ ПОЛИНОМА
        XRA     H               ;ВЫПОЛНИТЬ ОПЕРАЦИЮ ИСКЛЮЧАЮЩЕЕ ИЛИ
                                ; СО СТАРШИМ БАЙТОМ CRC
        MOV     H,A
```

```
CRCLP1:
        MOV     A,C             ;ВОССТАНОВИТЬ ДАННЫЕ
        RAL                     ;СДВИНУТЬ СЛУДУЮЩИЙ РАЗРЯД ДАННЫХ В
                                ; РАЗРЯД 7
        DCR     B               ;УМЕНЬШИТЬ СЧЕТЧИК РАЗРЯДОВ
        JNZ     CRCLP           ;ПЕРЕЙТИ, ЕСЛИ НЕ ПРОШЛИ ЧЕРЕЗ ВСЕ
                                ; 8 РАЗРЯДОВ
        SHLD    CRC             ;СОХРАНИТЬ ОБНОВЛЕННЫЙ CRC

        ;ВОССТАНОВИТЬ РЕГИСТРЫ И ВЫЙТИ
        POP     H
        POP     D
        POP     B
        POP     PSW
        RET


        ;*************************************
        ;ПОДПРОГРАММА: ICRC16
        ;НАЗНАЧЕНИЕ:   ИНИЦИАЛИЗИРУЕТ CRC И PLY
        ;ВХОД:  НЕТ ПАРАМЕТРОВ
        ;ВЫХОД: ИНИЦИАЛИЗИРУЮТСЯ CRC И ПОЛИНОМ
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: HL
        ;*************************************
ICRC16:
        LXI     H,0             ;CRC = 0
        SHLD    CRC
        LXI     H,08005H        ;PLY = 8005H
        SHLD    PLY
                                ;8005H ДЛЯ X^16+X^15+X^2+1 СОДЕРЖИТ
                                ;  1 В КАЖДОМ РАЗРЯДЕ, ДЛЯ КОТОРОГО
                                ;  В ФОРМУЛЕ ПОЯВЛЯЕТСЯ ВОЗВЕДЕНИЕ
                                ;  В СТЕПЕНЬ (РАЗРЯДЫ 0, 2 И 15)
        RET

        ;*************************************
        ;ПОДПРОГРАММА: GCRC16
        ;НАЗНАЧЕНИЕ:   ДАЕТ ЗНАЧЕНИЕ CRC
        ;ВХОД:  НЕТ ПАРАМЕТРОВ
        ;ВЫХОД: ПАРА РЕГИСТРОВ H = ЗНАЧЕНИЕ CRC
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: HL
        ;*************************************
GCRC16:
        LHLD    CRC             ;HL = CRC
        RET

        ;ДАННЫЕ
CRC:    DS      2               ;ЗНАЧЕНИЕ CRC
PLY:    DS      2               ;ЗНАЧЕНИЕ ПОЛИНОМА


;                                                               ;
;                                                               ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                       ;
;                                                               ;
;                                                               ;
```

```
        ;СГЕНЕРИРОВАТЬ CRC ДЛЯ ЧИСЛА 1 И ПРОВЕРИТЬ ЕГО
SC10C:
        CALL    ICRC16          ;ИНИЦИАЛИЗИРОВАТЬ CRC И ПОЛИНОМ
        MVI     A,1             ;СГЕНЕРИРОВАТЬ CRC ДЛЯ 1
        CALL    CRC16
        CALL    GCRC16
        XCHG                    ;СОХРАНИТЬ CRC В РЕГИСТРЕ DE
        CALL    ICRC16          ;СНОВА ИНИЦИАЛИЗИРОВАТЬ
        MVI     A,1
        CALL    CRC16           ;ПРОВЕРИТЬ ЗНАЧЕНИЕ CRC, СГЕНЕРИРОВАВ
                                ; ЕГО ДЛЯ ДАННЫХ
        MOV     A,D
        CALL    CRC16           ; И ЗАПОМНИТЬ CRC СНОВА
        MOV     A,E
        CALL    CRC16
        CALL    GCRC16          ;ЗНАЧЕНИЕ CRC В HL ДОЛЖНО  БЫТЬ
                                ; РАВНО НУЛЮ

        ;СГЕНЕРИРОВАТЬ CRC ДЛЯ ПОСЛЕДОВАТЕЛЬНОСТИ 0,1,2,...,255
        ; И ПРОВЕРИТЬ ЕГО ЗНАЧЕНИЕ
        CALL    ICRC16          ;ИНИЦИАЛИЗИРОВАТЬ CRC И ПОЛИНОМ
        MVI     B,0             ;НАЧАТЬ С БАЙТА ДАННЫХ 0
GENLP:
        MOV     A,B             ;ВЗЯТЬ БАЙТ ДАННЫХ
        CALL    CRC16           ;ОБНОВИТЬ CRC
        INR     B               ;ДОБАВИТЬ 1 ДЛЯ ПОЛУЧЕНИЯ СЛЕДУЮЩЕГО
                                ; БАЙТА ДАННЫХ
        JNZ     GENLP           ;ПЕРЕЙТИ, ЕСЛИ ЕЩЕ НЕ ВСЕ СДЕЛАНО
        CALL    GCRC16          ;ВЗЯТЬ РЕЗУЛЬТИРУЮЩЕЕ ЗНАЧЕНИЕ CRC
        XCHG                    ;И СОХРАНИТЬ ЕГО В DE

        ;ПРОВЕРИТЬ CRC, СГЕНЕРИРОВАВ ЕГО ЕЩЕ РАЗ
        CALL    ICRC16          ;ИНИЦИАЛИЗИРОВАТЬ CRC И ПОЛИНОМ
        MVI     B,0             ;НАЧАТЬ С БАЙТА ДАННЫХ 0
CHKLP:
        MOV     A,B             ;ВЗЯТЬ БАЙТ ДАННЫХ
        CALL    CRC16           ;ОБНОВИТЬ CRC
        INR     B               ;ДОБАВИТЬ 1 ДЛЯ ПОЛУЧЕНИЯ СЛЕДУЮЩЕГО
                                ; БАЙТА ДАННЫХ
        JNZ     CHKLP

        ;ВКЛЮЧИТЬ В ПРОВЕРКУ СОХРАНЕННОЕ ЗНАЧЕНИЕ CRC
        MOV     A,D             ;ВКЛЮЧИТЬ СТАРШИЙ БАЙТ СОХРАНЕННОГО
        CALL    CRC16           ; ЗНАЧЕНИЯ CRC

        MOV     A,E             ;ВКЛЮЧИТЬ МЛАДШИЙ БАЙТ СОХРАНЕННОГО
        CALL    CRC16           ; ЗНАЧЕНИЯ CRC

        CALL    GCRC16          ;ПОЛУЧИТЬ РЕЗУЛЬТИРУЮЩЕЕ ЗНАЧЕНИЕ CRC
                                ;ОНО ДОЛЖНО БЫТЬ РАВНО 0

        JMP     SC10C

        END
```

10D. ДИСПЕТЧЕР ТАБЛИЦЫ УСТРОЙСТВ ВВОДА-ВЫВОДА (IOHDLR)

Выполняется ввод и вывод способом, не зависящим от устройства с использованием блоков управления вводом-выводом и таблицы устройств ввода-вывода. Таблица устройств ввода-вывода представляет собой связанный список; каждая запись содержит ссылку на следующую запись, номер устройства и начальные адреса тех подпрограмм, которые инициализируют устройство, определяют его состояние ввода, читают данные с устройства, определяют его состояние вывода и записывают данные на устройство. Блок управления вводом-выводом — это массив, содержащий номер устройства, номер операции, данные о состоянии устройства и базовый адрес и длину буфера устройства. Если должен быть записан только один байт, пользователь должен задать подпрограмме IOHDLR базовый адрес блока управления вводом-выводом и данные. Подпрограмма IOHDLR возвращает байт состояния и данные (если должен был быть прочитан только один байт).

Эта подпрограмма является примером работы с вводом и выводом способом, не зависящим от устройства. Таблица устройств ввода-вывода должна создаваться с помощью подпрограмм INITDL, которая инициализирует список устройств, очищая его, и ADDDL, которая добавляет устройства к списку.

Прикладные программы могут выполнять ввод или вывод, получая или создавая блок управления вводом-выводом и затем обращаясь к IOHDLR. В подпрограмме IOHDLR используется таблица устройств ввода-вывода для определения способа передачи управления драйверу ввода-вывода.

Процедура. Сначала инициализируется байт состояния, в него записывается 0, что указывает на отсутствие ошибок. Затем просматривается таблица устройств в поисках номера устройства, записанного в блоке управления вводом-выводом. Если такого номера не находится, то осуществляется выход с номером ошибки в байте состояния. Если номер найден, то проверяется номер операции и управление передается соответствующей подпрограмме из записи таблицы устройств. Эта подпрограмма должна заканчиваться возвратом управления начально вызывавшей программе. Если операция неправильная (номер операции слишком большой или начальный адрес для подпрограммы равен 0), то осуществляется возврат с номером ошибки в байте состояния.

Подпрограмма INITDL инициализирует список устройств, устанавливая начальную связь равной 0.

Подпрограмма ADDDL добавляет запись к таблице устройств, делая ее базовый адрес началом списка и записывая в ее поле связи старое начало списка.

Используемые регистры:
1. IOHDLR: все.
2. INITDL: HL.
3. ADDDL: DE.

Время выполнения:
1. IOHDLR: 359 (8080) или 348 (8085) тактов плюс 93 (8080) или 87 (8085) тактов для каждого безуспешного сравнения номера устройства.
2. INITDL: 35 тактов (8080 или 8085).
3. ADDDL: 74 (8080) или 76 (8085) тактов.

Размер программы:

1. IOHDLR: 95 байт.
2. INITDL: 7 байт.
3. ADDDL: 13 байт.

Память, необходимая для данных: 5 байт в любом месте ОЗУ для базового адреса блока управления вводом-выводом (2 байта, начиная с адреса IOCBA), начала списка устройств (2 байта, начиная с адреса DVLST) и временного хранения данных, которые должны быть записаны без использования буфера (1 байт по адресу BDATA).

УСЛОВИЯ НА ВХОДЕ

1. IOHDLR: базовый адрес блока управления вводом-выводом в регистрах H и L.
2. INITDL: нет параметров.
3. ADDDL: базовый адрес таблицы устройств в регистрах H и L.

УСЛОВИЯ НА ВЫХОДЕ

1. IOHDLR: если найдена ошибка, в регистре A находится байт состояния блока ввода-вывода; в противном случае осуществляется выход из программы в соответствующий драйвер ввода-вывода. Если операция состоит в чтении одного байта, байт данных содержится в регистре A.
2. INITDL: начало списка устройств (адреса DVLST и DVLST+1), в которое записан 0, показывающий, что список пустой.
3. ADDDL: запись таблицы устройств добавляется в список.

ПРИМЕР

В примере, приведенном в листинге, используются следующие структуры:

Операции ввода-вывода

Номер операции	Описание операции
0	Инициализировать устройство
1	Определить состояние ввода
2	Прочитать 1 байт с устройства ввода
3	Прочитать N байт (обычно одну строку) с устройства ввода
4	Определить состояние вывода
5	Записать 1 байт на устройство вывода
6	Записать N байт (обычно одну строку) на устройство вывода

Блок управления вводом-выводом

Индекс	Содержание
0	Номер устройства
1	Номер операции
2	Состояние
3	Младший байт базового адреса буфера
4	Старший байт базового адреса буфера
5	Младший байт длины буфера
6	Старший байт длины буфера

Запись таблицы устройств

Индекс	Содержание
0	Младший байт поля связи (базового адреса следующей записи)
1	Старший байт поля связи (базового адреса следующей записи)
2	Номер устройства
3	Младший байт начального адреса подпрограммы инициализации устройства
4	Старший байт начального адреса подпрограммы инициализации устройства
5	Младший байт начального адреса подпрограммы определения состояния ввода
6	Старший байт начального адреса подпрограммы определения состояния ввода
7	Младший байт начального адреса драйвера ввода (только для чтения 1 байта)
8	Старший байт начального адреса драйвера ввода (только для чтения 1 байта)
9	Младший байт начального адреса драйвера ввода (N байт или строка)
10	Старший байт начального адреса драйвера ввода (N байт или строка)
11	Младший байт начального адреса подпрограммы определения состояния вывода
12	Старший байт начального адреса подпрограммы определения состояния вывода
13	Младший байт начального адреса драйвера вывода (только для записи 1 байта)
14	Старший байт начального адреса драйвера вывода (только для записи 1 байта)
15	Младший байт начального адреса драйвера вывода (N байт или строка)
16	Старший байт начального адреса драйвера вывода (N байт или строка)

Если операция недопустима или неопределенна (например, определение состояния вывода для клавиатуры или драйвер ввода для устройства печати), соответствующий начальный адрес в таблице равен 0.

Значения для состояния

Значение	Описание
0	Нет ошибок
1	Неправильный номер устройства (нет такого устройства)
2	Неправильный номер операции (нет такой операции или ошибочная операция)
3	Вводимые данные доступны или устройство вывода готово
254	Буфер слишком мал для использования функции 10 BDOS CP/M (считать с консоли в буфер). Эта функция требует 2 байта для длины буфера и счетчика символов

```
;                                                                  ;
;                                                                  ;
;                                                                  ;
;                                                                  ;
;       ЗАГОЛОВОК:      ДИСПЕТЧЕР ТАБЛИЦЫ УСТРОЙСТВ ВВОДА-ВЫВОДА   ;
;       ИМЯ:            IOHDLR                                     ;
;                                                                  ;
```

```
;                                                                    ;
;                                                                    ;
;       НАЗНАЧЕНИЕ:     ВЫПОЛНЯЕТ НЕ ЗАВИСЯЩИЙ ОТ УСТРОЙСТВ ВВОД-ВЫВОД. ;
;                       ЭТО МОЖНО СДЕЛАТЬ ТОЛЬКО С ПОМОЩЬЮ ОРГАНИЗАЦИИ  ;
;                       ДОСТУПА КО ВСЕМ УСТРОЙСТВАМ ОДНИМ И ТЕМ ЖЕ      ;
;                       СПОСОБОМ, ИСПОЛЬЗУЯ БЛОК УПРАВЛЕНИЯ ВВОДОМ-     ;
;                       ВЫВОДОМ (IOCB) И ТАБЛИЦУ УСТРОЙСТВ.             ;
;                       ПРЕДСТАВЛЕННЫЕ ЗДЕСЬ ПОДПРОГРАММЫ ПОЗВОЛЯЮТ     ;
;                       ОСУЩЕСТВИТЬ СЛЕДУЮЩИЕ ОПЕРАЦИИ:                 ;
;                                                                    ;
;                       НОМЕР ОПЕРАЦИИ          ОПИСАНИЕ                ;
;                           0           ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТВО     ;
;                           1           ОПРЕДЕЛИТЬ СОСТОЯНИЕ ВВОДА      ;
;                           2           ПРОЧИТАТЬ 1 БАЙТ                ;
;                           3           ПРОЧИТАТЬ N БАЙТ                ;
;                           4           ОПРЕДЕЛИТЬ СОСТОЯНИЕ ВЫВОДА     ;
;                           5           ЗАПИСАТЬ 1 БАЙТ                 ;
;                           6           ЗАПИСАТЬ N БАЙТ                 ;
;                                                                    ;
;                       ДРУГИЕ ОПЕРАЦИИ, ТАКИЕ КАК ОТКРЫТЬ, ЗАКРЫТЬ,    ;
;                       УДАЛИТЬ, ПЕРЕИМЕНОВАТЬ И ДОПОЛНИТЬ, КОТОРЫЕ     ;
;                       МОЖНО БЫЛО БЫ СЮДА ВКЛЮЧИТЬ, МОГЛИ БЫ РАБОТАТЬ  ;
;                       С ТАКИМИ УСТРОЙСТВАМИ, КАК ГИБКИЕ ДИСКИ         ;
;                                                                    ;
;                       IOCB - ЭТО МАССИВ, ИМЕЮЩИЙ СЛЕДУЮЩИЙ ФОРМАТ:    ;
;                                                                    ;
;                       IOCB + 0 = НОМЕР УСТРОЙСТВА                     ;
;                       IOCB + 1 = НОМЕР ОПЕРАЦИИ                       ;
;                       IOCB + 2 = СОСТОЯНИЕ                            ;
;                       IOCB + 3 = МЛАДШИЙ БАЙТ АДРЕСА БУФЕРА           ;
;                       IOCB + 4 = СТАРШИЙ БАЙТ АДРЕСА БУФЕРА           ;
;                       IOCB + 5 = МЛАДШИЙ БАЙТ ДЛИНЫ БУФЕРА            ;
;                       IOCB + 6 = СТАРШИЙ БАЙТ ДЛИНЫ БУФЕРА            ;
;                                                                    ;
;                       ТАБЛИЦА УСТРОЙСТВ ВЫПОЛНЕНА В ВИДЕ СВЯЗАННОГО   ;
;                       СПИСКА. СО СПИСКОМ РАБОТАЮТ ДВЕ ПОДПРОГРАММЫ:   ;
;                       INITDL, КОТОРАЯ ИНИЦИАЛИЗИРУЕТ СПИСОК, ОЧИЩАЯ   ;
;                       ЕГО, И ADDDL, КОТОРАЯ ДОБАВЛЯЕТ В СПИСОК        ;
;                       УСТРОЙСТВА. ЗАПИСЬ ТАБЛИЦЫ УСТРОЙСТВ ИМЕЕТ      ;
;                       СЛЕДУЮЩИЙ ФОРМАТ:                               ;
;                                                                    ;
;                       DVTBL + 0 = МЛАДШИЙ БАЙТ ПОЛЯ СВЯЗИ             ;
;                       DVTBL + 1 = СТАРШИЙ БАЙТ ПОЛЯ СВЯЗИ             ;
;                       DVTBL + 2 = НОМЕР УСТРОЙСТВА                    ;
;                       DVTBL + 3 = МЛАДШИЙ БАЙТ ПОДПРОГРАММЫ           ;
;                                   ИНИЦИАЛИЗАЦИИ УСТРОЙСТВА            ;
;                       DVTBL + 4 = СТАРШИЙ БАЙТ ПОДПРОГРАММЫ           ;
;                                   ИНИЦИАЛИЗАЦИИ УСТРОЙСТВА            ;
;                       DVTBL + 5 = МЛАДШИЙ БАЙТ ПОДПРОГРАММЫ СОСТОЯНИЯ ;
;                                   ВВОДА                               ;
;                       DVTBL + 6 = СТАРШИЙ БАЙТ ПОДПРОГРАММЫ СОСТОЯНИЯ ;
;                                   ВВОДА                               ;
;                       DVTBL + 7 = МЛАДШИЙ БАЙТ ПОДПРОГРАММЫ ВВОДА     ;
;                                   1 БАЙТА                             ;
;                       DVTBL + 8 = СТАРШИЙ БАЙТ ПОДПРОГРАММЫ ВВОДА     ;
;                                   1 БАЙТА                             ;
;                       DVTBL + 9 = МЛАДШИЙ БАЙТ ПОДПРОГРАММЫ ВВОДА     ;
```

```
;                                     N БАЙТ                                  ;
;                           DVTBL + 10= СТАРШИЙ БАЙТ ПОДПРОГРАММЫ ВВОДА       ;
;                                     N БАЙТ                                  ;
;                           DVTBL + 11= МЛАДШИЙ БАЙТ ПОДПРОГРАММЫ СОСТОЯНИЯ   ;
;                                     ВЫВОДА                                  ;
;                           DVTBL + 12= СТАРШИЙ БАЙТ ПОДПРОГРАММЫ СОСТОЯНИЯ   ;
;                                     ВЫВОДА                                  ;
;                           DVTBL + 13= МЛАДШИЙ БАЙТ ПОДПРОГРАММЫ ВЫВОДА      ;
;                                     1 БАЙТА                                 ;
;                           DVTBL + 14= СТАРШИЙ БАЙТ ПОДПРОГРАММЫ ВЫВОДА      ;
;                                     1 БАЙТА                                 ;
;                           DVTBL + 15= МЛАДШИЙ БАЙТ ПОДПРОГРАММЫ ВЫВОДА      ;
;                                     N БАЙТ                                  ;
;                           DVTBL + 16= СТАРШИЙ БАЙТ ПОДПРОГРАММЫ ВЫВОДА      ;
;                                     N БАЙТ                                  ;
;                                                                             ;
;       ВХОД:               ПАРА РЕГИСТРОВ HL = БАЗОВЫЙ АДРЕС IOCB            ;
;                           РЕГИСТР A = ДАННЫЕ ДЛЯ ЗАПИСИ 1 БАЙТА (БУФЕР      ;
;                                     НЕ ИСПОЛЬЗУЕТСЯ)                        ;
;                                                                             ;
;       ВЫХОД:              РЕГИСТР A = КОПИЯ БАЙТА СОСТОЯНИЯ ИЗ IOCB, ИЛИ    ;
;                                     ДАННЫЕ ПРИ ЧТЕНИИ 1 БАЙТА (БУФЕР        ;
;                                     НЕ ИСПОЛЬЗУЕТСЯ).                       ;
;                           БАЙТ СОСТОЯНИЯ IOCB СОДЕРЖИТ 0, ЕСЛИ ОПЕРАЦИЯ     ;
;                           ЗАКОНЧИЛАСЬ УСПЕШНО; В ПРОТИВНОМ СЛУЧАЕ ОН        ;
;                           СОДЕРЖИТ НОМЕР ОШИБКИ.                            ;
;                                                                             ;
;                           ЗНАЧЕНИЯ СОСТОЯНИЯ           ОПИСАНИЕ             ;
;                               0                НЕТ ОШИБОК                   ;
;                               1                НЕПРАВИЛЬНЫЙ НОМЕР УСТРОЙСТВА;
;                               2                НЕПРАВИЛЬНЫЙ НОМЕР ОПЕРАЦИИ  ;
;                               3                ВВОДИМЫЕ ДАННЫЕ ДОСТУПНЫ ИЛИ ;
;                                                УСТРОЙСТВО ВЫВОДА ГОТОВО     ;
;                               254              БУФЕР СЛИШКОМ МАЛ ДЛЯ ФУНКЦИИ;
;                                                10 BDOS CP/M (ПРОЧИТАТЬ БУФЕР;
;                                                С КОНСОЛИ)                   ;
;                                                                             ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ:  ВСЕ                                           ;
;                                                                             ;
;       ВРЕМЯ:              МИНИМАЛЬНОЕ 359 ТАКТОВ ПЛЮС 93 ТАКТА НА КАЖДОЕ    ;
;                           УСТРОЙСТВО В СПИСКЕ КРОМЕ ТОГО, КОТОРОЕ БЫЛО      ;
;                           ЗАПРОШЕНО В ДАННОМ ОБРАЩЕНИИ, ДЛЯ 8080            ;
;                           МИНИМАЛЬНОЕ 348 ТАКТОВ ПЛЮС 87 ТАКТОВ НА КАЖДОЕ   ;
;                           УСТРОЙСТВО В СПИСКЕ КРОМЕ ТОГО, КОТОРОЕ БЫЛО      ;
;                           ЗАПРОШЕНО В ДАННОМ ОБРАЩЕНИИ, ДЛЯ 8085            ;
;                                                                             ;
;       РАЗМЕР:             ПРОГРАММА - 115 БАЙТ                              ;
;                           ДАННЫЕ    -   5 БАЙТ                              ;
;                                                                             ;
;                                                                             ;

        ;ЭКВИВАЛЕНТНЫЕ ЗНАЧЕНИЯ ДЛЯ IOCB И ТАБЛИЦЫ УСТРОЙСТВ
IOCBDN  EQU     0       ;НОМЕР УСТРОЙСТВА IOCB
IOCBOP  EQU     1       ;НОМЕР ОПЕРАЦИИ IOCB
IOCBST  EQU     2       ;СОСТОЯНИЕ IOCB
IOCBBA  EQU     3       ;АДРЕС БУФЕРА IOCB
IOCBBL  EQU     5       ;ДЛИНА БУФЕРА IOCB
```

```
DTLNK   EQU     0          ;ПОЛЕ СВЯЗИ ТАБЛИЦЫ УСТРОЙСТВ
DTDN    EQU     2          ;НОМЕР УСТРОЙСТВА ТАБЛИЦЫ УСТРОЙСТВ
DTSR    EQU     3          ;НАЧАЛО ПОДПРОГРАММ ТАБЛИЦЫ УСТРОЙСТВ
        ;НОМЕРА ОПЕРАЦИЙ
NUMOP   EQU     7          ;ЧИСЛО ОПЕРАЦИЙ
INIT    EQU     0          ;ИНИЦИАЛИЗАЦИЯ
ISTAT   EQU     1          ;СОСТОЯНИЕ ВВОДА
R1BYTE  EQU     2          ;ПРОЧИТАТЬ 1 БАЙТ
RNBYTE  EQU     3          ;ПРОЧИТАТЬ N БАЙТ
OSTAT   EQU     4          ;СОСТОЯНИЕ ВЫВОДА
W1BYTE  EQU     5          ;ЗАПИСАТЬ 1 БАЙТ
WNBYTE  EQU     6          ;ЗАПИСАТЬ N БАЙТ

        ;ЗНАЧЕНИЯ СОСТОЯНИЯ
NOERR   EQU     0          ;НЕТ ОШИБОК
DEVERR  EQU     1          ;НЕПРАВИЛЬНЫЙ НОМЕР УСТРОЙСТВА
OPERR   EQU     2          ;НЕПРАВИЛЬНЫЙ НОМЕР ОПЕРАЦИИ
DEVRDY  EQU     3          ;ВХОДНЫЕ ДАННЫЕ ДОСТУПНЫ ИЛИ УСТРОЙСТВО
                           ; ВЫВОДА ГОТОВО
BUFERR  EQU     254        ;БУФЕР СЛИШКОМ МАЛ ДЛЯ ФУНКЦИИ BDOS ЧТЕНИЯ
                           ; БУФЕРА С КОНСОЛИ

IOHDLR:
        ;СОХРАНИТЬ IOCB И ДАННЫЕ (ЕСЛИ ОНИ ЕСТЬ)
        SHLD    IOCBA               ;СОХРАНИТЬ АДРЕС IOCB
        XCHG                        ;DE = АДРЕС IOCB
        STA     BDATA               ;СОХРАНИТЬ БАЙТ ДАННЫХ ДЛЯ ЗАПИСИ 1 БАЙТА

        ;ЗАДАТЬ НАЧАЛЬНОЕ ЗНАЧЕНИЕ БАЙТА СОСТОЯНИЯ - НЕТ ОШИБОК
        LXI     H,IOCBST            ;ВЗЯТЬ АДРЕС БАЙТА СОСТОЯНИЯ
        DAD     D
        MVI     M,NOERR             ;СОСТОЯНИЕ = НЕТ ОШИБОК

        ;ПРОВЕРИТЬ НОМЕР ОПЕРАЦИИ (ПОПАДАЕТ ЛИ В ДОПУСТИМЫЕ ПРЕДЕЛЫ)
        LXI     H,IOCBOP            ;ВЗЯТЬ НОМЕР ОПЕРАЦИИ ИЗ IOCB
        DAD     D
        MOV     A,M
        MOV     B,A                 ;СОХРАНИТЬ НОМЕР ОПЕРАЦИИ
        CPI     NUMOP               ;НОМЕР ОПЕРАЦИИ В ДОПУСТИМЫХ ПРЕДЕЛАХ?
        JNC     BADOP               ;ВЫЙТИ, ЕСЛИ НОМЕР ОПЕРАЦИИ СЛИШКОМ ВЕЛИК

        ;НАЙТИ В СПИСКЕ ЗАПИСЬ ДЛЯ ДАННОГО УСТРОЙСТВА
        ;DE = УКАЗАТЕЛЬ НА СПИСОК УСТРОЙСТВ
        ;C = НОМЕР УСТРОЙСТВА IOCB
        LXI     H,IOCBDN            ;ВЗЯТЬ АДРЕС НОМЕРА УСТРОЙСТВА В IOCB
        DAD     D
        MOV     C,M                 ;C = НОМЕР УСТРОЙСТВА IOCB
        LHLD    DVLST               ;DE = ПЕРВАЯ ЗАПИСЬ В СПИСКЕ УСТРОЙСТВ
        XCHG

        ;DE = УКАЗАТЕЛЬ НА СПИСОК УСТРОЙСТВ
        ;B = НОМЕР ОПЕРАЦИИ
        ;C = НОМЕР ЗАПРОШЕННОГО УСТРОЙСТВА
SRCHLP:
        ;ПРОВЕРИТЬ НА КОНЕЦ СПИСКА УСТРОЙСТВ (ПОЛЕ СВЯЗИ = 0000)
        MOV     A,D                 ;ПРОВЕРИТЬ ПОЛЕ СВЯЗИ
        ORA     E
```

```
        JZ      BADDN               ;ПЕРЕЙТИ, ЕСЛИ НЕТ БОЛЬШЕ ЗАПИСЕЙ ДЛЯ
                                    ; УСТРОЙСТВ

        ;ПРОВЕРИТЬ, ОТНОСИТСЯ ЛИ ДАННАЯ ЗАПИСЬ К ЗАПРОШЕННОМУ УСТРОЙСТВУ
        LXI     H,DTDN              ;ВЗЯТЬ АДРЕС НОМЕРА УСТРОЙСТВА В ЗАПИСИ
        DAD     D
        MOV     A,M                 ;СРАВНИТЬ С ЗАПРОШЕННЫМ УСТРОЙСТВОМ
        CMP     C
        JZ      FOUND               ;ПЕРЕЙТИ, ЕСЛИ УСТРОЙСТВО НАЙДЕНО

        ;УСТРОЙСТВО НЕ НАЙДЕНО, ТОГДА, ИСПОЛЬЗУЯ ПОЛЕ СВЯЗИ, ПЕРЕЙТИ
        ; К СЛЕДУЮЩЕЙ ЗАПИСИ ТАБЛИЦЫ УСТРОЙСТВ, ВЗЯВ АДРЕС ЗАПИСИ ДЛЯ
        ; ТЕКУЩЕГО УСТРОЙСТВА ИЗ ПОЛЯ СВЯЗИ
        XCHG                        ;ПОЛУЧИТЬ АДРЕС ПОЛЯ СВЯЗИ
                                    ; (ПЕРВОЕ СЛОВО)
        MOV     E,M                 ;ВЗЯТЬ МЛАДШИЙ БАЙТ СВЯЗИ
        INX     H
        MOV     D,M                 ;ВЗЯТЬ СТАРШИЙ БАЙТ СВЯЗИ
        JMP     SRCHLP              ;ПРОВЕРИТЬ СЛЕДУЮЩУЮ ЗАПИСЬ В ТАБЛИЦЕ
                                    ; УСТРОЙСТВ

        ;УСТРОЙСТВО НАЙДЕНО, ТОГДА ПЕРЕДАТЬ УПРАВЛЕНИЕ СООТВЕТСТВУЮЩЕЙ
        ; ПОДПРОГРАММЕ, ЕСЛИ ОНА ЕСТЬ
        ;DE = АДРЕС ЗАПИСИ ТАБЛИЦЫ УСТРОЙСТВ
        ;B = НОМЕР ОПЕРАЦИИ В IOCB
FOUND:
        ;ВЗЯТЬ АДРЕС ПОДПРОГРАММЫ (НУЛЬ УКАЗЫВАЕТ НА НЕПРАВИЛЬНУЮ
        ; ОПЕРАЦИЮ)
        MOV     L,B                 ;HL = 16-РАЗРЯДНЫЙ НОМЕР ОПЕРАЦИИ
        MVI     H,0
        DAD     H                   ;УМНОЖИТЬ НА 2 ДЛЯ АДРЕСА
        LXI     B,DTSR
        DAD     B                   ;HL = СМЕЩЕНИЕ ДЛЯ АДРЕСА ПОДПРОГРАММЫ
                                    ; В ЗАПИСИ ТАБЛИЦЫ УСТРОЙСТВ
        DAD     D                   ;HL УКАЗЫВАЕТ НА АДРЕС ПОДПРОГРАММЫ
        MOV     A,M                 ;ВЗЯТЬ НАЧАЛЬНЫЙ АДРЕС ПОДПРОГРАММЫ
        INX     H
        MOV     H,M
        MOV     L,A                 ;ПРОВЕРИТЬ НАЧАЛЬНЫЙ АДРЕС НА НУЛЬ
        ORA     H
        JZ      BADOP               ;ПЕРЕЙТИ, ЕСЛИ ОПЕРАЦИЯ НЕПРАВИЛЬНАЯ
                                    ; (АДРЕС = 0)
        PUSH    H                   ;ЗАПИСАТЬ В СТЕК АДРЕС ПОДПРОГРАММЫ
        LHLD    IOCBA               ;HL = БАЗОВЫЙ АДРЕС IOCB
        LDA     BDATA               ;A = БАЙТ ДАННЫХ ДЛЯ ЗАПИСИ 1 БАЙТА
        RET                         ;ПЕРЕЙТИ НА ПОДПРОГРАММУ

BADDN:
        MVI     A,DEVERR            ;КОД ОШИБКИ -- НЕТ ТАКОГО УСТРОЙСТВА
        JMP     EREXIT

BADOP:
        MVI     A,OPERR             ;КОД ОШИБКИ -- НЕТ ТАКОЙ ОПЕРАЦИИ

EREXIT:
        LHLD    IOCBA               ;ВЗЯТЬ АДРЕС БАЙТА СОСТОЯНИЯ IOCB
        LXI     D,IOCBST
```

```
        DAD     D
        MOV     M,A                     ;ПОСЛАТЬ БАЙТ СОСТОЯНИЯ В IOCB
        RET


        ;****************************************
        ;ПОДПРОГРАММА: INITDL
        ;НАЗНАЧЕНИЕ:   ИНИЦИАЛИЗИРОВАТЬ СПИСОК УСТРОЙСТВ,
        ;              СДЕЛАВ ЕГО ПУСТЫМ
        ;ВХОД:  НЕТ ПАРАМЕТРОВ
        ;ВЫХОД: УСТАНАВЛИВАЕТ ОТСУТСТВИЕ ЗАПИСЕЙ В СПИСКЕ УСТРОЙСТВ
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: HL
        ;****************************************
INITDL:
        ;ЗАПИСАТЬ 0 В ЗАГОЛОВОК СПИСКА УСТРОЙСТВ, ЧТОБЫ УКАЗАТЬ НА

        ; ОТСУТСТВИЕ УСТРОЙСТВ
        LXI     H,0                     ;ЗАГОЛОВОК = 0 (ПУСТОЙ СПИСОК)
        SHLD    DVLST
        RET

        ;****************************************
        ;ПОДПРОГРАММА: ADDDL
        ;НАЗНАЧЕНИЕ:   ДОБАВИТЬ УСТРОЙСТВО В СПИСОК УСТРОЙСТВ
        ;ВХОД:  ПАРА РЕГИСТРОВ H = АДРЕС ЗАПИСИ ТАБЛИЦЫ УСТРОЙСТВ
        ;ВЫХОД: УСТРОЙСТВО ДОБАВЛЯЕТСЯ В СПИСОК УСТРОЙСТВ
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: DE
        ;****************************************
ADDDL:
        XCHG                            ;ВЗЯТЬ ТЕКУЩИЙ ЗАГОЛОВОК СПИСКА
        LHLD    DVLST                   ; УСТРОЙСТВ
        XCHG                            ; В DE
                                        ;HL = АДРЕС НОВОГО УСТРОЙСТВА
        MOV     M,E                     ;ЗАПОМНИТЬ ТЕКУЩИЙ ЗАГОЛОВОК СПИСКА
        INX     H                       ; УСТРОЙСТВ В ПОЛЕ СВЯЗИ ЗАПИСИ ДЛЯ
        MOV     M,D                     ; НОВОГО УСТРОЙСТВА
        DCX     H
        SHLD    DVLST                   ;СДЕЛАТЬ DVLST УКАЗАТЕЛЕМ НА ЗАПИСИ
                                        ; ДЛЯ НОВОГО УСТРОЙСТВА
        RET

        ;СЕКЦИЯ ДАННЫХ
IOCBA:  DS      2                       ;БАЗОВЫЙ АДРЕС IOCB
DVLST:  DS      2                       ;ЗАГОЛОВОК СПИСКА УСТРОЙСТВ
BDATA:  DS      1                       ;БАЙТ ДАННЫХ ДЛЯ ЗАПИСИ 1 БАЙТА


;                                                                       ;
;                                                                       ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                               ;
;                                                                       ;
;         В ДАННОЙ ТЕСТОВОЙ ПРОГРАММЕ УСТРОЙСТВОМ 1 НАЗНАЧАЕТСЯ         ;
;       КОНСОЛЬ CP/M, А УСТРОЙСТВОМ 2 - УСТРОЙСТВО ПЕЧАТИ CP/M.         ;
;       ЗАТЕМ ПРОГРАММА ЧИТАЕТ СТРОКУ С КОНСОЛИ И ВЫДАЕТ ЕЕ НА          ;
;       КОНСОЛЬ И УСТРОЙСТВО ПЕЧАТИ.                                    ;
;                                                                       ;
;                                                                       ;
```

```
        ;ЭКВИВАЛЕНТНЫЕ СИМВОЛЫ
CR      EQU     0DH             ;СИМВОЛ ВОЗВРАТА КАРЕТКИ
LF      EQU     0AH             ;СИМВОЛ ПЕРЕВОДА СТРОКИ

        ;ЭКВИВАЛЕНТНЫЕ ЗНАЧЕНИЯ CP/M
BDOS    EQU     0005H           ;АДРЕС ТОЧКИ ВХОДА BDOS CP/M
CINP    EQU     1               ;ФУНКЦИЯ BDOS ВВОДА С КОНСОЛИ
COUTP   EQU     2               ;ФУНКЦИЯ BDOS ВЫВОДА НА КОНСОЛЬ
LOUTP   EQU     5               ;ФУНКЦИЯ BDOS ВЫВОДА НА ПЕЧАТЬ
RCBUF   EQU     10              ;ФУНКЦИЯ BDOS ВВОДА БУФЕРА С КОНСОЛИ
CSTAT   EQU     11              ;ФУНКЦИЯ BDOS СОСТОЯНИЯ КОНСОЛИ

SC10D:
        ;ИНИЦИАЛИЗИРОВАТЬ СПИСОК УСТРОЙСТВ
        CALL    INITDL          ;СПИСОК УСТРОЙСТВ ПУСТОЙ

        ;НАЗНАЧИТЬ В КАЧЕСТВЕ УСТРОЙСТВА 1 КОНСОЛЬ И
        ; ИНИЦИАЛИЗИРОВАТЬ ЕЕ
        LXI     H,CONDV         ;ВЗЯТЬ АДРЕС ЗАПИСИ УСТРОЙСТВА ДЛЯ
                                ; КОНСОЛИ
        CALL    ADDDL           ;ДОБАВИТЬ КОНСОЛЬ В СПИСОК УСТРОЙСТВ
        MVI     A,INIT          ;ОПЕРАЦИЯ ИНИЦИАЛИЗАЦИИ
        STA     IOCB+IOCBOP
        MVI     A,1             ;НОМЕР УСТРОЙСТВА = 1
        STA     IOCB+IOCBDN
        LXI     H,IOCB          ;ИНИЦИАЛИЗИРОВАТЬ КОНСОЛЬ
        CALL    IOHDLR

        ;НАЗНАЧИТЬ В КАЧЕСТВЕ УСТРОЙСТВА 2 УСТРОЙСТВО ПЕЧАТИ
        ; И ИНИЦИАЛИЗИРОВАТЬ ЕГО
        LXI     H,PRTDV         ;ВЗЯТЬ АДРЕС ЗАПИСИ ДЛЯ УСТРОЙСТВА
                                ; ПЕЧАТИ
        CALL    ADDDL           ;ДОБАВИТЬ УСТРОЙСТВО ПЕЧАТИ В СПИСОК
                                ; УСТРОЙСТВ
        MVI     A,INIT          ;ОПЕРАЦИЯ ИНИЦИАЛИЗАЦИИ
        STA     IOCB+IOCBOP
        MVI     A,2             ;НОМЕР УСТРОЙСТВА = 2
        STA     IOCB+IOCBDN
        LXI     H,IOCB          ;ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТВО ПЕЧАТИ
        CALL    IOHDLR

        ;ЦИКЛ ЧТЕНИЯ СТРОК С КОНСОЛИ И ВЫДАЧИ ИХ НА КОНСОЛЬ И ПЕЧАТЬ;
        ; РАБОТА В ЦИКЛЕ ПРОДОЛЖАЕТСЯ ДО ТЕХ ПОР, ПОКА НЕ БУДЕТ ВВЕДЕНА
        ; ПУСТАЯ СТРОКА
TSTLP:
        MVI     A,1             ;НОМЕР УСТРОЙСТВА = 1 (КОНСОЛЬ)
        STA     IOCB+IOCBDN
        MVI     A,RNBYTE        ;ОПЕРАЦИЯ - ЧТЕНИЕ N БАЙТ
        STA     IOCB+IOCBOP
        LXI     H,LENBUF
        SHLD    IOCB+IOCBBL     ;УСТАНОВИТЬ ДЛИНУ БУФЕРА РАВНОЙ LENBUF
        LXI     H,IOCB          ;HL = БАЗОВЫЙ АДРЕС IOCB
        CALL    IOHDLR          ;ПРОЧИТАТЬ СТРОКУ С КОНСОЛИ

        ;ВЫВЕСТИ НА КОНСОЛЬ ПЕРЕВОД СТРОКИ
        MVI     A,W1BYTE        ;ОПЕРАЦИЯ - ЗАПИСЬ 1 БАЙТА
        STA     IOCB+IOCBOP
```

```
        MVI     A,LF                ;СИМВОЛ - ПЕРЕВОД СТРОКИ
        LXI     H,IOCB              ;HL = БАЗОВЫЙ АДРЕС IOCB
        CALL    IOHDLR              ;ЗАПИСАТЬ 1 БАЙТ (ПЕРЕВОД СТРОКИ)

        ;ВЫДАТЬ СТРОКУ НА УСТРОЙСТВА 1 И 2
        MVI     A,1
        CALL    ECHO                ;ВЫДАТЬ СТРОКУ НА УСТРОЙСТВО 1
        MVI     A,2
        CALL    ECHO                ;ВЫДАТЬ СТРОКУ НА УСТРОЙСТВО 2

        ;ЗАКОНЧИТЬ, ЕСЛИ ДЛИНА СТРОКИ РАВНА 0
        LHLD    IOCB+IOCBBL         ;ВЗЯТЬ ДЛИНУ СТРОКИ
        MOV     A,H                 ;ПРОВЕРИТЬ ДЛИНУ СТРОКИ
        ORA     L
        JNZ     TSTLP               ;ПЕРЕЙТИ, ЕСЛИ ДЛИНА НЕ РАВНА НУЛЮ

        JMP     SC10D               ;ВЫПОЛНИТЬ ТЕСТ ЕЩЕ РАЗ

ECHO:
        ;ВЫВЕСТИ СТРОКУ
        STA     IOCB+IOCBDN         ;УСТАНОВИТЬ НОМЕР УСТРОЙСТВА В IOCB
                                    ;ЗАМЕТИМ, ЧТО ПОДПРОГРАММА ECHO
                                    ; ПОСЫЛАЕТ СТРОКУ НА ЛЮБОЕ УСТРОЙСТВО.
                                    ; НОМЕР УСТРОЙСТВА НАХОДИТСЯ
                                    ; В АККУМУЛЯТОРЕ
        MVI     A,WNBYTE            ;УСТАНОВИТЬ ОПЕРАЦИЮ ДЛЯ ЗАПИСИ N БАЙТ
        STA     IOCB+IOCBOP
        LXI     H,IOCB              ;HL = БАЗОВЫЙ АДРЕС IOCB
        CALL    IOHDLR              ;ЗАПИСАТЬ N БАЙТ
        ;ВЫВЕСТИ СИМВОЛЫ ВОЗВРАТА КАРЕТКИ И ПЕРЕВОДА СТРОКИ
        MVI     A,W1BYTE            ;УСТАНОВИТЬ ОПЕРАЦИЮ ДЛЯ ЗАПИСИ 1 БАЙТА
        STA     IOCB+IOCBOP
        MVI     A,CR                ;СИМВОЛ - ВОЗВРАТ КАРЕТКИ
        LXI     H,IOCB              ;HL = БАЗОВЫЙ АДРЕС IOCB
        CALL    IOHDLR               ЗАПИСАТЬ 1 БАЙТ
        MVI     A,LF                ;СИМВОЛ - ПЕРЕВОД СТРОКИ
        LXI     H,IOCB              ;HL = БАЗОВЫЙ АДРЕС IOCB
        CALL    IOHDLR              ;ЗАПИСАТЬ 1 БАЙТ
        RET

        ;IOCB ДЛЯ ВЫПОЛНЕНИЯ ВВОДА-ВЫВОДА
IOCB:   DS      1                   ;НОМЕР УСТРОЙСТВА
        DS      1                   ;НОМЕР ОПЕРАЦИИ
        DS      1                   ;СОСТОЯНИЕ
        DW      BUFFER              ;АДРЕС БУФЕРА
        DS      2                   ;ДЛИНА БУФЕРА

        ;БУФЕР
LENBUF  EQU     127
BUFFER: DS      LENBUF

        ;ЗАПИСИ ТАБЛИЦЫ УСТРОЙСТВ
CONDV:  DW      0                   ;ПОЛЕ СВЯЗИ
        DB      1                   ;УСТРОЙСТВО 1
        DW      CINIT               ;ИНИЦИАЛИЗАЦИЯ КОНСОЛИ
        DW      CISTAT              ;СОСТОЯНИЕ ВВОДА С КОНСОЛИ
        DW      CIN                 ;ВВЕСТИ С КОНСОЛИ 1 БАЙТ
```

```
        DW      CINN                    ;ВВЕСТИ С КОНСОЛИ N БАЙТ
        DW      COSTAT                  ;СОСТОЯНИЕ ВЫВОДА НА КОНСОЛЬ
        DW      COUT                    ;ВЫВЕСТИ НА КОНСОЛЬ 1 БАЙТ
        DW      COUTN                   ;ВЫВЕСТИ НА КОНСОЛЬ N БАЙТ

PRTDV:  DW      0                       ;ПОЛЕ СВЯЗИ
        DB      2                       ;УСТРОЙСТВО 2
        DW      PINIT                   ;ИНИЦИАЛИЗАЦИЯ КОНСОЛИ
        DW      0                       ;ДЛЯ ПЕЧАТИ НЕТ СОСТОЯНИЯ ВВОДА
        DW      0                       ;ДЛЯ ПЕЧАТИ НЕТ ВВОДА 1 БАЙТА
        DW      0                       ;ДЛЯ ПЕЧАТИ НЕТ ВВОДА N БАЙТ
        DW      POSTAT                  ;СОСТОЯНИЕ ВЫВОДА НА ПЕЧАТЬ
        DW      POUT                    ;ВЫВЕСТИ НА ПЕЧАТЬ 1 БАЙТ
        DW      POUTN                   ;ВЫВЕСТИ НА ПЕЧАТЬ N БАЙТ

        ;************************************
        ;ПОДПРОГРАММЫ ВВОДА-ВЫВОДА ДЛЯ КОНСОЛИ
        ;************************************

        ;ИНИЦИАЛИЗАЦИЯ КОНСОЛИ
CINIT:
        SUB     A                       ;СОСТОЯНИЕ = НЕТ ОШИБКИ
        RET                             ;ИНИЦИАЛИЗАЦИЯ НЕ НУЖНА

        ;СОСТОЯНИЕ ВВОДА С КОНСОЛИ

CISTAT:
        PUSH    H                       ;СОХРАНИТЬ АДРЕС IOCB
        MVI     C,CSTAT                 ;ФУНКЦИЯ BDOS СОСТОЯНИЯ КОНСОЛИ
        CALL    BDOS                    ;ВЗЯТЬ СОСТОЯНИЕ КОНСОЛИ
        POP     H                       ;ВОССТАНОВИТЬ АДРЕС IOCB
        ORA     A                       ;ПРОВЕРИТЬ СОСТОЯНИЕ КОНСОЛИ
        JZ      CIS1                    ;ПЕРЕЙТИ, ЕСЛИ СИМВОЛ НЕ ГОТОВ (A = 0)
        MVI     A,DEVRDY                ;УКАЗАТЬ, ЧТО СИМВОЛ ГОТОВ
CIS1:   LXI     D,IOCBST                ;ЗАПОМНИТЬ СОСТОЯНИЕ В IOCB
        DAD     D
        MOV     M,A
        RET

        ;ЧТЕНИЕ С КОНСОЛИ 1 БАЙТА
CIN:
        MVI     C,CINP                  ;ФУНКЦИЯ BDOS ВВОДА С КОНСОЛИ
        CALL    BDOS                    ;СЧИТАТЬ 1 БАЙТ С КОНСОЛИ
        RET

        ;ЧТЕНИЕ С КОНСОЛИ N БАЙТ
CINN:
        ;ПРОЧИТАТЬ СТРОКУ, ИСПОЛЬЗУЯ ФУНКЦИЮ BDOS ЧТЕНИЯ С КОНСОЛИ
        ; В БУФЕР
        ;ФУНКЦИЯ BDOS ЧТЕНИЯ С КОНСОЛИ В БУФЕР ИСПОЛЬЗУЕТ СЛЕДУЮЩИЙ
        ; ФОРМАТ БУФЕРА:
        ;  БАЙТ 0: ДЛИНА БУФЕРА (МАКСИМАЛЬНОЕ ЧИСЛО СИМВОЛОВ)
        ;  БАЙТ 1: ЧИСЛО ПРОЧИТАННЫХ СИМВОЛОВ (ДЛИНА СТРОКИ)
        ;  БАЙТЫ 2 И ДАЛЕЕ: ПРОЧИТАННЫЕ СИМВОЛЫ
        XCHG                            ;DE = БАЗОВЫЙ АДРЕС IOCB
        LXI     H,IOCBBL                ;ВЗЯТЬ ДЛИНУ БУФЕРА
        DAD     D
```

```
        MOV     A,M                 ;A = ДЛИНА БУФЕРА
        SBI     3                   ;БУФЕР ДОЛЖЕН СОСТОЯТЬ ПО  КРАЙНЕЙ МЕРЕ
                                    ; ИЗ 3-Х СИМВОЛОВ ДЛЯ МАКСИМАЛЬНОЙ
                                    ; ДЛИНЫ И СЧЕТЧИКА, ИСПОЛЬЗУЕМЫХ BDOS
                                    ; ДЛЯ ЧТЕНИЯ С КОНСОЛИ В БУФЕР
        JNC     CINN1               ;ПЕРЕЙТИ, ЕСЛИ БУФЕР ИМЕЕТ ДОСТАТОЧНУЮ
                                    ; ДЛИНУ
        LXI     H,IOCBST            ;УСТАНОВИТЬ СОСТОЯНИЕ ОШИБКИ - БУФЕР
        DAD     D                   ; СЛИШКОМ МАЛ, НЕТ МЕСТА ДЛЯ ДАННЫХ
        MVI     M,BUFERR
        RET

CINN1:  INR     A                   ;ДОБАВИТЬ ЕДИНИЦУ НАЗАД, ЧТОБЫ
                                    ; НАЙТИ ЧИСЛО БАЙТОВ В БУФЕРЕ
                                    ; ДЛЯ ДАННЫХ (ЗАГОЛОВОК ИЗ 2-Х БАЙТ)
        PUSH    H                   ;СОХРАНИТЬ АДРЕС ДЛИНЫ БУФЕРА В IOCB
        LXI     H,IOCBBA            ;ВЗЯТЬ ИЗ IOCB АДРЕС БУФЕРА
        DAD     D
        MOV     E,M                 ;ВЗЯТЬ МЛАДШИЙ БАЙТ АДРЕСА БУФЕРА
        INX     H
        MOV     D,M                 ;ВЗЯТЬ СТАРШИЙ БАЙТ АДРЕСА БУФЕРА
        PUSH    D                   ;СОХРАНИТЬ АДРЕС БУФЕРА
        STAX    D                   ;УСТАНОВИТЬ МАКСИМАЛЬНУЮ ДЛИНУ БУФЕРА
        MVI     C,RCBUF             ;ФУНКЦИЯ BDOS ЧТЕНИЯ С КОНСОЛИ В БУФЕР
        CALL    BDOS                ;ПРОЧИТАТЬ В БУФЕР

        ;ВЕРНУТЬ В IOCB ЧИСЛО ПРОЧИТАННЫХ СИМВОЛОВ
        POP     H                   ;ВОССТАНОВИТЬ АДРЕС БУФЕРА
        INX     H                   ;ВЗЯТЬ ЧИСЛО ПРОЧИТАННЫХ СИМВОЛОВ
        MOV     A,M
        XCHG
        POP     H                   ;ВЗЯТЬ АДРЕС ДЛИНЫ БУФЕРА В IOCB
        MOV     M,A                 ;УСТАНОВИТЬ В IOCB ДЛИНУ БУФЕРА
        INX     H
        MVI     M,0                 ;СТАРШИЙ БАЙТ ДЛИНЫ РАВЕН 0

        ;ПЕРЕСЫЛАТЬ ДАННЫЕ В БУФЕР, НАЧИНАЯ С ПЕРВОГО БАЙТА, ЗАТИРАЯ
        ; ТАКИМ ОБРАЗОМ ЗАГОЛОВОК БУФЕРА (ДЛИНУ БУФЕРА, ЧИСЛО
        ; ПРОЧИТАННЫХ СИМВОЛОВ), ВОЗВРАЩАЕМЫЙ CP/M. ДЛИНА СТРОКИ
        ; ТЕПЕРЬ НАХОДИТСЯ В IOCB
        ORA     A                   ;ПРОВЕРИТЬ ЧИСЛО ПРОЧИТАННЫХ
                                    ; СИМВОЛОВ
        RZ                          ;ЕСЛИ НЕТ СИМВОЛОВ, ВЕРНУТЬСЯ
        MOV     B,A                 ;B = ЧИСЛО ПРОЧИТАННЫХ СИМВОЛОВ
        MOV     H,D
        MOV     L,E
        INX     H                   ;HL = ПЕРВЫЙ БАЙТ ДАННЫХ, НАХОДЯЩИЙСЯ
                                    ; ЧЕРЕЗ 2 БАЙТА ПОСЛЕ НАЧАЛА
        DCX     D                   ;DE = АДРЕС НАЗНАЧЕНИЯ (ПЕРВЫЙ БАЙТ
                                    ; БУФЕРА)

CIMVLP:
        MOV     A,M                 ;ПЕРЕСЛАТЬ СЛЕДУЮЩИЙ БАЙТ, НАХОДЯЩИЙСЯ
        STAX    D                   ; В БУФЕРЕ НА 2 БАЙТА НИЖЕ
        INX     H                   ;УВЕЛИЧИТЬ АДРЕС ИСТОЧНИКА
        INX     D                   ;УВЕЛИЧИТЬ АДРЕС НАЗНАЧЕНИЯ
        DCR     B                   ;УМЕНЬШИТЬ СЧЕТЧИК
```

```
        JNZ     CIMVLP              ;ПРОДОЛЖАТЬ, ПОКА НЕ БУДУТ ПЕРЕСЛАНЫ
                                    ; ВСЕ СИМВОЛЫ
        SUB     A                   ;ВОЗВРАТИТЬСЯ, НЕТ ОШИБОК
        RET

        ;СОСТОЯНИЕ ВЫВОДА НА КОНСОЛЬ
COSTAT:
        MVI     A,DEVRDY            ;СОСТОЯНИЕ - КОНСОЛЬ ВСЕГДА ГОТОВА
                                    ; К ВЫВОДУ
        RET

        ;ВЫВОД НА КОНСОЛЬ 1 БАЙТА
COUT:
        MVI     C,2                 ;ОПЕРАЦИЯ BDOS ВЫВОДА НА КОНСОЛЬ
        MOV     E,A                 ;E = СИМВОЛ
        CALL    BDOS                ;ВЫВЕСТИ 1 БАЙТ
        SUB     A                   ;СОСТОЯНИЕ = НЕТ ОШИБОК
        RET

        ;ВЫВОД НА КОНСОЛЬ N БАЙТ
COUTN:
        LXI     D,COUT              ;DE = АДРЕС ПОДПРОГРАММЫ ВЫВОДА СИМВОЛА
        CALL    OUTN                ;ВЫЗВАТЬ ВЫВОД N СИМВОЛОВ
        SUB     A                   ;СОТОЯНИЕ = НЕТ ОШИБОК
        RET

        ;*************************************
        ;ПОДПРОГРАММЫ ДЛЯ УСТРОЙСТВА ПЕЧАТИ
        ;*************************************

        ;ИНИЦИАЛИЗАЦИЯ УСТРОЙСТВА ПЕЧАТИ

PINIT:
        SUB     A                   ;НИЧЕГО НЕ ДЕЛАТЬ, ВЕРНУТЬСЯ БЕЗ ОШИБОК
        RET

        ;СОСТОЯНИЕ ВЫВОДА НА УСТРОЙСТВО ПЕЧАТИ
POSTAT:
        MVI     A,DEVRDY            ;СОСТОЯНИЕ - УСТРОЙСТВО ВСЕГДА ГОТОВО
        RET

        ;ВЫВЕСТИ НА ПЕЧАТЬ 1 БАЙТ
POUT:
        MVI     C,LOUTP             ;ФУНКЦИЯ BDOS ВЫВОДА НА ПЕЧАТЬ
        MOV     E,A                 ;E = СИМВОЛ
        CALL    BDOS                ;ВЫВЕСТИ НА ПЕЧАТЬ
        SUB     A                   ;СОСТОЯНИЕ = НЕТ ОШИБОК
        RET
        ;ВЫВЕСТИ НА ПЕЧАТЬ N БАЙТ
POUTN:
        LXI     D,POUT              ;DE = АДРЕС ПОДПРОГРАММЫ ВЫВОДА
        CALL    OUTN                ;ВЫВЕСТИ N СИМВОЛОВ
        SUB     A                   ;СОСТОЯНИЕ = НЕТ ОШИБОК
        RET

        ;*************************************
        ;ПОДПРОГРАММА: OUTN
        ;НАЗНАЧЕНИЕ:   ВЫВЕСТИ N СИМВОЛОВ
```

```
        ;ВХОД:  РЕГИСТР DE = АДРЕС ПОДПРОГРАММЫ ВЫВОДА СИМВОЛА
        ;       РЕГИСТР HL = БАЗОВЫЙ АДРЕС IOCB
        ;ВЫХОД: ВЫВОД ДАННЫХ
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ
        ;****************************************
OUTN:
        ;ЗАПОМНИТЬ АДРЕС ПОДПРОГРАММЫ ВЫВОДА СИМВОЛА
        XCHG
        SHLD    COSR            ;СОХРАНИТЬ АДРЕС ПОДПРОГРАММЫ ВЫВОДА

        ;ВЗЯТЬ ЧИСЛО БАЙТОВ; ЕСЛИ ДЛИНА РАВНА 0, ВЫЙТИ
        ; BC = ЧИСЛО БАЙТОВ
        LXI     H,IOCBBL        ;ВЗЯТЬ АДРЕС ДЛИНЫ БУФЕРА В IOCB
        DAD     D
        MOV     C,M             ;ВЗЯТЬ МЛАДШИЙ БАЙТ ДЛИНЫ БУФЕРА
        INX     H
        MOV     B,M             ;И СТАРШИЙ БАЙТ ДЛИНЫ БУФЕРА
        MOV     A,B             ;ПРОВЕРИТЬ ДЛИНУ БУФЕРА
        ORA     C
        RZ                      ;ВЫЙТИ, ЕСЛИ БУФЕР ПУСТОЙ

        ;ВЗЯТЬ ИЗ IOCB АДРЕС БУФЕРА ВЫВОДА
        ; HL = АДРЕС БУФЕРА
        LXI     H,IOCBBA        ;ВЗЯТЬ АДРЕС БУФЕРА В IOCB
        DAD     D
        MOV     A,M             ;ВЗЯТЬ МЛАДШИЙ БАЙТ АДРЕСА БУФЕРА
        INX     H
        MOV     H,M             ;И СТАРШИЙ БАЙТ АДРЕСА БУФЕРА
        MOV     L,A             ;HL = АДРЕС БУФЕРА

OUTLP:
        MOV     A,M             ;ВЗЯТЬ ДАННЫЕ ИЗ БУФЕРА
        PUSH    H               ;СОХРАНИТЬ УКАЗАТЕЛЬ В БУФЕРЕ,
        PUSH    B               ; СЧЕТЧИК СИМВОЛОВ
        CALL    DOSUB           ;ВЫВЕСТИ СИМВОЛ
        POP     B               ;ВОССТАНОВИТЬ УКАЗАТЕЛЬ, СЧЕТЧИК
        POP     H
        INX     H               ;УКАЗАТЬ НА СЛЕДУЮЩИЙ СИМВОЛ
        DCX     B               ;УМЕНЬШИТЬ И ПРОВЕРИТЬ СЧЕТЧИК
        MOV     A,B
        ORA     C
        JNZ     OUTLP           ;ПРОДОЛЖАТЬ, ПОКА СЧЕТЧИК НЕ БУДЕТ = 0
        RET

DOSUB:  LHLD    COSR
        PCHL                    ;ПЕРЕЙТИ К ПОДПРОГРАММЕ

COSR:   DW      0               ;АДРЕС ПОДПРОГРАММЫ ВЫВОДА СИМВОЛА

        END
```

10Е. ИНИЦИАЛИЗАЦИЯ ПОРТОВ ВВОДА-ВЫВОДА (IPORTS)

Инициализируется набор портов ввода-вывода, заданный массивом адресов портов устройств и значений данных. В приведенных примерах показана инициализация распространенных программируемых устройств ввода-вывода 8080, 8085: 8251 PCI, 8253 PIT и 8255 PPI.

В подпрограмме реализуется обобщенный метод для секций инициализации ввода-вывода. Могут инициализироваться порты данных, регистры, определяющие направление потока данных и назначение отдельных разрядов для ввода или вывода, управляющие или командные регистры, назначающие режимы работы программируемых устройств, счетчики (в таймерах), регистры приоритетов и другие внешние регистры или запоминающие ячейки.

Эта подпрограмма является самомодифицируемой, так как должны изменяться адреса устройств в командах OUT. Необходимость в этом возникает вследствие того, что команда OUT допускает только прямую адресацию (устройство), поэтому нет простого способа выводить данные на устройства с различными адресами.

С помощью этой подпрограммы пользователь может выполнять следующие задачи:

1) назначать двунаправленные линии в качестве линий ввода или вывода;
2) инициализировать порты вывода;
3) разрешать или запрещать прерывания от периферийных устройств;
4) задавать способы работы, как, например: фиксировать ли введенные данные, выдавать ли стробирующие импульсы, должен ли таймер работать постоянно или только по запросам, и тому подобное;
5) загружать начальные значения таймеров и счетчиков;
6) выбирать скорости для передачи;
7) очищать или выдавать команду сброса на устройства, не связанные с общей линией сброса;
8) инициализировать регистры приоритетов или назначать начальные приоритеты для прерываний или других операций;
9) инициализировать векторы при обслуживании прерываний, прямого доступа к памяти и других операций.

Процедура. Для каждого порта получается число байтов, которое необходимо послать, и адрес устройства. Затем модифицируется специально выделенная команда OUT, используемая для пересылки значений данных в порт. При таком подходе достигается независимость от числа или типов устройств в секции ввода-вывода. Пользователь может добавлять или удалять устройства или изменять процесс инициализации, изменяя лишь массив, а не программу.

Каждая запись массива содержит набор элементов длиной в байт, расположенных в следующем порядке:

1) число байтов, которые должны посылаться в порт;
2) 8-разрядный адрес устройства для порта;
3) байты данных в порядке их выдачи.

Этот массив заканчивается терминатором, который содержит 0 в первом байте.

Отметим, что запись может содержать произвольное число байтов. Первый элемент определяет, сколько байтов необходимо послать на устройство, адрес которого задан во втором элементе. Последующие элементы содержат значения данных. Терминатору требуется всего лишь один байт, содержащий 0.

Фактическая команда OUT находится в подпрограмме в памяти для данных. Перед передачей управления подпрограмме адрес устройства из таблицы записывается во второй байт этой команды.

Используемые регистры: все.
Время выполнения: 22 (8080) или 23 (8085) такта плюс 61 + 57 * N тактов для каждого порта, где N – число посылаемых байтов.
Размер программы: 25 байт плюс размер таблицы (по крайней мере, по 3 байта на порт плюс 1 байт на терминатор).
Память, необходимая для данных: отсутствует, однако подпрограмма OUTPUT (3 байта) должна обязательно находиться в ОЗУ, так как программа модифицирует адрес порта в команде этой подпрограммы.

УСЛОВИЯ НА ВХОДЕ

Базовый адрес массива инициализации в регистрах H и L.

УСЛОВИЯ НА ВЫХОДЕ

Все значения данных посылаются в соответствующие порты.

ПРИМЕР

1. Данные: массив содержит следующие элементы:
   3 (число байтов для порта 1); адрес устройства порта 1, 1-е – 3-е значения,
   2 (число байтов для порта 2); адрес устройства порта 2, 1-е, 2-е значения,
   4 (число байтов для порта 3); адрес устройства порта 3, 1-е – 4-е значения,
   0 (терминатор).

   Результат: по адресу 1-го порта посылаются три значения,
   по адресу 2-го порта посылаются два значения,
   по адресу 3-го порта посылаются четыре значения.

```
;                                                                  ;
;                                                                  ;
;                                                                  ;
;                                                                  ;
;       ЗАГОЛОВОК:      ИНИЦИАЛИЗАЦИЯ ПОРТОВ ВВОДА-ВЫВОДА          ;
;       ИМЯ:            IPORTS                                     ;
;                                                                  ;
;                                                                  ;


;                                                                  ;
;       НАЗНАЧЕНИЕ:     ИНИЦИАЛИЗИРУЕТ ПОРТЫ ВВОДА-ВЫВОДА, ЗАДАННЫЕ В ;
;                       МАССИВЕ АДРЕСОВ ПОРТОВ И ЗНАЧЕНИЙ ДЛЯ      ;
;                       ИНИЦИАЛИЗАЦИИ                              ;
;                                                                  ;
;       ВХОД:           ПАРА РЕГИСТРОВ H = БАЗОВЫЙ АДРЕС МАССИВА   ;
;                                                                  ;
;                       МАССИВ СОДЕРЖИТ ЭЛЕМЕНТЫ ДЛИНОЙ В БАЙТ,    ;
;                       РАСПОЛОЖЕННЫЕ В СЛЕДУЮЩЕМ ПОРЯДКЕ: ЧИСЛО   ;
;                       БАЙТОВ, КОТОРОЕ НЕОБХОДИМО ПОСЛАТЬ В ПОРТ, АДРЕС;
;                       ПОРТА УСТРОЙСТВА, ЗНАЧЕНИЯ ДАННЫХ ДЛЯ ПОРТА. ;
;                       ЭТА ПОСЛЕДОВАТЕЛЬНОСТЬ ПОВТОРЯЕТСЯ ДЛЯ ЛЮБОГО ;
;                       ЧИСЛА ПОРТОВ. МАССИВ ЗАКАНЧИВАЕТСЯ ЗАПИСЬЮ, ;
;                       СОДЕРЖАЩЕЙ 0 В ЧИСЛЕ БАЙТОВ.               ;
;                          МАССИВ+0 = ЧИСЛО БАЙТОВ ДЛЯ ДАННОГО ПОРТА ;
;                          МАССИВ+1 = АДРЕС ПОРТА УСТРОЙСТВА       ;
;                          МАССИВ+2 = ПЕРВОЕ ЗНАЧЕНИЕ ДЛЯ ДАННОГО ПОРТА ;
;                                     .                            ;
;                                     .                            ;
```

```
;                                   .                                       ;
;                       МАССИВ+2+(N-1) = ПОСЛЕДНЕЕ ЗНАЧЕНИЕ ДЛЯ             ;
;                                        ДАННОГО ПОРТА                      ;
;                                   .                                       ;
;                                   .                                       ;
;                                   .                                       ;
;                                                                           ;
;       ВЫХОД:          НЕТ ПАРАМЕТРОВ                                      ;
;                                                                           ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ:  ВСЕ                                         ;
;                                                                           ;
;       ВРЕМЯ:          22 ТАКТА ПЛЮС 61 + (N * 57) ТАКТОВ ДЛЯ КАЖДОГО      ;
;                       ПОРТА, ГДЕ N - ЧИСЛО ПОСЫЛАЕМЫХ БАЙТОВ, ДЛЯ         ;
;                       8080                                                ;
;                       23 ТАКТА ПЛЮС 61 + (N * 57) ТАКТОВ ДЛЯ КАЖДОГО      ;
;                       ПОРТА, ГДЕ N - ЧИСЛО ПОСЫЛАЕМЫХ БАЙТОВ, ДЛЯ         ;
;                       8085                                                ;
;                                                                           ;
;       РАЗМЕР:         ПРОГРАММА - 25 БАЙТ                                 ;
;                                                                           ;
;                                                                           ;

IPORTS:
        ;ВЗЯТЬ ЧИСЛО БАЙТОВ ДАННЫХ, КОТОРОЕ НЕОБХОДИМО ПОСЛАТЬ В
        ; ТЕКУЩИЙ ПОРТ
        ;ВЫЙТИ, ЕСЛИ ЧИСЛО БАЙТОВ РАВНО 0, ЧТО УКАЗЫВАЕТ НА ТЕРМИНАТОР
        MOV     A,M             ;ВЗЯТЬ ЧИСЛО БАЙТОР
        ORA     A               ;ПРОВЕРИТЬ НА РАВЕНСТВО  НУЛЮ
                                ; (ТЕРМИНАТОР)
        RZ                      ;ВЕРНУТЬСЯ, ЕСЛИ ЧИСЛО БАЙТОВ = 0
        MOV     B,A
        INX     H               ;УКАЗАТЬ НА АДРЕС ПОРТА (СЛЕДУЮЩИЙ БАЙТ)

        ;ВЗЯТЬ АДРЕС ПОРТА И ПОСЛАТЬ ЕГО ВО ВТОРОЙ БАЙТ (АДРЕС
        ; УСТРОЙСТВА) КОМАНДЫ OUT. ЭТО МОДИФИКАЦИЯ КОМАНДЫ, ПОЭТОМУ
        ; КОМАНДА OUT ДОЛЖНА НАХОДИТЬСЯ В ОЗУ


        MOV     A,M             ;ВЗЯТЬ АДРЕС ПОРТА
        STA     PORTAD          ;ЗАПОМНИТЬ ЕГО В БАЙТЕ АДРЕСА
                                ; КОМАНДЫ OUT 0
        INX     H               ;УКАЗАТЬ НА ПЕРВОЕ ЗНАЧЕНИЕ ДАННЫХ
                                ; (СЛЕДУЮЩИЙ БАЙТ)

        ;ВЫВЕСТИ ДАННЫЕ И ПРОДОЛЖИТЬ ДЛЯ СЛЕДУЮЩЕГО ПОРТА
IPLP:
        MOV     A,M             ;ВЗЯТЬ СЛЕДУЮЩИЙ БАЙТ ДАННЫХ
        CALL    OUTPUT          ;ВЫВЕСТИ ДАННЫЕ В ПОРТ
        INX     H               ;УВЕЛИЧИТЬ ДЛЯ СЛЕДУЮЩЕГО БАЙТА
        DCR     B               ;УМЕНЬШИТЬ ЧИСЛО БАЙТОВ
        JNZ     IPLP            ;ПРОДОЛЖАТЬ, ПОКА ВСЕ НЕ БУДЕТ
                                ; ВЫПОЛНЕНО

        JMP     IPORTS          ;ПРОДОЛЖИТЬ ДЛЯ СЛЕДУЮЩЕГО ПОРТА

        ;СЕКЦИЯ ДАННЫХ
OUTPUT:
PORTAD  EQU     OUTPUT+1        ;АДРЕС НОМЕРА ПОРТА В ДАННОЙ НИЖЕ
                                ; КОМАНДЕ ВЫВОДА
```

```
        OUT     0                       ;КОМАНДА ВЫВОДА (0 - ПУСТОЙ АДРЕС)
        RET


;                                                                              ;
;                                                                              ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                                      ;
;                                                                              ;
;                                                                              ;

        ;ИНИЦИАЛИЗИРОВАТЬ:
        ;  8253 PIT (ПРОГРАММИРУЕМЫЙ ТАЙМЕР)
        ;  ПОСЛЕДОВАТЕЛЬНЫЙ ИНТЕРФЕЙС 8251 (ПРОГРАММИРУЕМЫЙ ИНТЕРФЕЙС
        ;   СВЯЗИ, ИЛИ PCI)
        ;  ПАРАЛЛЕЛЬНЫЙ ИНТЕРФЕЙС 8255 (ПРОГРАММИРУЕМЫЙ ПЕРИФЕРИЙНЫЙ
        ;   ИНТЕРФЕЙС, ИЛИ PPI)

        ;ПРОИЗВОЛЬНЫЕ АДРЕСА ПОРТОВ
        ;
        ; АДРЕСА 8253 PIT
PIT0    EQU     70H                     ;КАНАЛ 0 8253
PIT1    EQU     71H                     ;КАНАЛ 1 8253
PIT2    EQU     72H                     ;КАНАЛ 2 8253
PITMDE  EQU     73H                     ;СЛОВО РЕЖИМА РАБОТЫ 8253

        ; АДРЕСА 8251 PCI
PCID    EQU     80H                     ;ПОРТ ДАННЫХ 8251
PCIC    EQU     81H                     ;ПОРТ СОСТОЯНИЯ/УПРАВЛЕНИЯ 8251

        ; АДРЕСА 8255 PPI
PPIPA   EQU     0C0H                    ;ПОРТ A 8255
PPIPB   EQU     0C1H                    ;ПОРТ B 8255
PPIPC   EQU     0C2H                    ;ПОРТ C 8255
PPIC    EQU     0C3H                    ;ПОРТ УПРАВЛЕНИЯ 8255

SC10E:
        LXI     H,PINIT                 ;ВЗЯТЬ АДРЕС МАССИВА ИНИЦИАЛИЗАЦИИ
        CALL    IPORTS                  ;ИНИЦИАЛИЗИРОВАТЬ ПОРТЫ
        IN      PCID                    ;ВЫПОЛНИТЬ ХОЛОСТОЕ ЧТЕНИЕ ДЛЯ
                                        ; УВЕРЕННОСТИ В ТОМ, ЧТО 8251
                                        ; НАХОДИТСЯ В ПРАВИЛЬНОМ СОСТОЯНИИ

        JMP     SC10E

PINIT:
        ;ИНИЦИАЛИЗИРОВАТЬ СЧЕТЧИК 0 8253 PIT
        ;
        ;    ЗАДАТЬ СЧЕТЧИК ТАКИМ ОБРАЗОМ, ЧТОБЫ ОН ГЕНЕРИРОВАЛ
        ;     ПОСЛЕДОВАТЕЛЬНОСТЬ ПРЯМОУГОЛЬНЫХ ИМПУЛЬСОВ, УМЕНЬШАЯ
        ;     СЧЕТЧИК НА КАЖДОМ ОТРИЦАТЕЛЬНОМ (СНИЖАЮЩЕМСЯ) ФРОНТЕ
        ;     ЗАДАЮЩЕГО ИМПУЛЬСА. ПЕРИОД ПОСЛЕДОВАТЕЛЬНОСТИ ПРЯМОУГОЛЬНЫХ
        ;     ИМПУЛЬСОВ РАВЕН НАЧАЛЬНОМУ ЗНАЧЕНИЮ, ЗАГРУЖЕННОМУ В
        ;     СЧЕТЧИК.
        ;    ВЫПОЛНЯТЬ ДВОИЧНЫЙ СЧЕТ, УСТАНОВИВ НАЧАЛЬНОЕ ЗНАЧЕНИЕ РАВНЫМ
        ;     ДЕСЯТИЧНОМУ ЧИСЛУ 13.
        ;    ЗАМЕЧАНИЕ: ПОСЛЕ ГЕНЕРАЦИИ КАЖДОЙ ПОСЛЕДОВАТЕЛЬНОСТИ
        ;     ПРЯМОУГОЛЬНЫХ ИМПУЛЬСОВ 8253 СНОВА ЗАГРУЖАЕТ В СЧЕТЧИК
        ;     НАЧАЛЬНОЕ ЗНАЧЕНИЕ.
```

```
;    ПРИМЕЧАНИЕ: ЕСЛИ НАЧАЛЬНОЕ ЗНАЧЕНИЕ НЕЧЕТНОЕ, ИМПУЛЬСЫ В
;     ДЕЙСТВИТЕЛЬНОСТИ НЕ ПРЯМОУГОЛЬНЫЕ. НА ВЫВОДЕ 10 ТАЙМЕРА
;     8253 ВЫХОДНОЙ УРОВЕНЬ БУДЕТ ВЫСОКИМ ДЛЯ ЗНАЧЕНИЙ СЧЕТЧИКА
;     (N+1)/2 И НИЗКИМ ДЛЯ ЗНАЧЕНИЙ СЧЕТЧИКА (N-1)/2, ГДЕ N -
;     НАЧАЛЬНОЕ ЗНАЧЕНИЕ. В ДАННОМ СЛУЧАЕ ВЫХОДНОЙ УРОВЕНЬ БУДЕТ
;     ВЫСОКИМ ДЛЯ 7 ЗНАЧЕНИЙ СЧЕТЧИКА И НИЗКИМ - ДЛЯ 6 ЗНАЧЕНИЙ.

;    ЭТА ИНИЦИАЛИЗАЦИЯ ЗАДАЕТ ИМПУЛЬСЫ ДЛЯ РАБОТЫ 8251 PCI СО
;     СКОРОСТЬЮ ПЕРЕДАЧИ 9600 БОД.
;    СЧИТАЕТСЯ, ЧТО НА ВЫВОД 9 ТАКТОВЫЕ ИМПУЛЬСЫ ПРИХОДЯТ С
;     ЧАСТОТОЙ 2 МГЦ, ПОЭТОМУ ЗНАЧЕНИЕ СЧЕТЧИКА 2000000/153600 =
;     13 БУДЕТ ГЕНЕРИРОВАТЬ ПОСЛЕДОВАТЕЛЬНОСТЬ ПРЯМОУГОЛЬНЫХ
;     ИМПУЛЬСОВ НА ВЫВОДЕ 9 PCI (ИМПУЛЬСЫ ПЕРЕДАЧИ) И ВЫВОДЕ 25
;     (ИМПУЛЬСЫ ПРИЕМА) С ЧАСТОТОЙ 153600 (16*9600) ГЦ.
;     PCI РАБОТАЕТ В РЕЖИМЕ ДЕЛЕНИЯ НА 16 ИМПУЛЬСОВ.
DB      1               ;ВЫВОД ОДНОГО БАЙТА
DB      PITMDE          ;ПОРТ НАЗНАЧЕНИЯ - РЕГИСТР РЕЖИМА РАБОТЫ
DB      00110110B       ;РАЗРЯД 0 = 0 (ДВОИЧНЫЙ СЧЕТ)
                        ;РАЗРЯДЫ 3..1 = 011 (РЕЖИМ РАБОТЫ 3 -
                        ; ГЕНЕРАТОР СКОРОСТИ ПОСЛЕДОВАТЕЛЬНОСТИ
                        ; ПРЯМОУГОЛЬНЫХ ИМПУЛЬСОВ)
                        ;РАЗРЯДЫ 5,4 = 11 (ЗАГРУЗИТЬ В СЧЕТЧИК
                        ; 2 БАЙТА)
                        ;РАЗРЯДЫ 7,6 = 00 (ВЫБРАТЬ СЧЕТЧИК 0)
DB      2               ;ВЫВОД 2 БАЙТ
DB      PIT0            ;ПОРТ НАЗНАЧЕНИЯ - СЧЕТЧИК 0
DB      13              ;МЛАДШИЙ БАЙТ СЧЕТЧИКА
DB      0               ;СТАРШИЙ БАЙТ СЧЕТЧИКА

;ИНИЦИАЛИЗИРОВАТЬ 8251 ДЛЯ АСИНХРОННОГО ПОСЛЕДОВАТЕЛЬНОГО
; ВВОДА-ВЫВОДА.
;       СБРОСИТЬ ПРОГРАММНО 8251, ПОСЛАВ 2 БАЙТА 80Н
;        С ПОСЛЕДУЮЩИМ БАЙТОМ 40Н.
;       УСТАНОВИТЬ АСИНХРОННЫЙ РЕЖИМ РАБОТЫ, 8-РАЗРЯДНЫЕ СИМВОЛЫ,
;        БЕЗ РАЗРЯДА ЧЕТНОСТИ, 2 СТОПОВЫХ РАЗРЯДА, УМНОЖЕНИЕ
;        ЧАСТОТЫ ИМПУЛЬСОВ НА 16.
;       РАЗРЕШИТЬ РАБОТУ ПРИЕМНИКА И ПЕРЕДАТЧИКА, СБРОСИТЬ
;        ИНДИКАТОРЫ ОШИБОК
DB      4               ;ВЫВОД 4 БАЙТ
DB      PCIC            ;ПОРТ НАЗНАЧЕНИЯ - УПРАВЛЯЮЩИЙ ПОРТ PCI
                        ;КОМАНДА НАСТРОЙКИ НА АСИНХРОННЫЙ РЕЖИМ
                        ; РАБОТЫ
DB      80H             ;СБРОСИТЬ 8251
DB      80H             ; С ПОМОЩЬЮ ДВУХ БАЙТ 80Н,
DB      40H             ; ЗА КОТОРЫМИ СЛЕДУЕТ КОМАНДА СБРОСА
DB      11001110B       ;РАЗРЯДЫ 1,0 = 10 (КОЭФФИЦИЕНТ УМНОЖЕНИЯ
                        ; СКОРОСТИ НА 16)
                        ;РАЗРЯДЫ 3,2 = 11 (8-РАЗРЯДНЫЙ СИМВОЛ)
                        ;РАЗРЯД 4 = 0 (БЕЗ РАЗРЯДА ЧЕТНОСТИ)
                        ;РАЗРЯД 5 = 0 (ЗНАЧЕНИЕ БЕЗРАЗЛИЧНО)
                        ;РАЗРЯДЫ 7,6 = 11 (2 СТОПОВЫХ РАЗРЯДА)

                        ;КОМАНДА АСИНХРОННОЙ РАБОТЫ
DB      00010111B       ;РАЗРЯД 0 = 1 (РАЗРЕШИТЬ ПЕРЕДАЧУ)
                        ;РАЗРЯД 1 = 1 (ДАННЫЕ НА ТЕРМИНАЛЕ
                        ; ГОТОВЫ
                        ;РАЗРЯД 2 = 1 (РАЗРЕШИТЬ ПРИЕМ)
```

```
                              ;РАЗРЯД 3 = 0 (НЕТ СИМВОЛА BREAK
                              ; (ПРЕРВАТЬ))
                              ;РАЗРЯД 4 = 1 (СБРОСИТЬ ОШИБКУ)
                              ;РАЗРЯД 5 = 0 (РАЗРЕШИТЬ ПЕРЕДАЧУ)
                              ;РАЗРЯД 6 = 0 (НЕТ ЗАПРОСА, КОТОРЫЙ
                              ; ТРЕБУЕТСЯ ПОСЛАТЬ)
                              ;РАЗРЯД 7 = 0 (НЕТ ПРОИЗВОЛЬНОГО ПОИСКА)

;ИНИЦИАЛИЗИРОВАТЬ 8255 PPI (ПАРАЛЛЕЛЬНЫЙ ИНТЕРФЕЙС)
;
;       ПОРТ A - ВВОД, ПОРТ B - ВЫВОД, ВЕРХНИЕ 4 РАЗРЯДА ПОРТА
;        C - ВЫВОД, НИЖНИЕ 4 РАЗРЯДА ПОРТА C - ВВОД, БЕЗ
;        АВТОМАТИЧЕСКИХ СИГНАЛОВ ПОДТВЕРЖДЕНИЯ (РЕЖИМ 0 -
;        ОСНОВНОЙ ВВОД-ВЫВОД)
DB      1               ;ВЫВОД 1-ГО БАЙТА
DB      PPIC            ;ПОРТ НАЗНАЧЕНИЯ - УПРАВЛЯЮЩИЙ ПОРТ PPI
DB      10010001B       ;РАЗРЯД 0 = 1 (НИЖНИЙ ПОРТ C - ВВОД)
                        ;РАЗРЯД 1 = 0 (ПОРТ B - ВЫВОД)
                        ;РАЗРЯД 2 = 0 (РЕЖИМ 0 ПОРТА B)
                        ;РАЗРЯД 3 = 0 (ВЕРХНИЙ ПОРТ C - ВЫВОД)
                        ;РАЗРЯД 4 = 1 (ПОРТ A - ВВОД)
                        ;РАЗРЯДЫ 6,5 = 00 (РЕЖИМ 0 ПОРТА A)
                        ;РАЗРЯД 7 = 1 (КОМАНДА ВЫБОРА РЕЖИМА)

;КОНЕЦ ДАННЫХ ДЛЯ ИНИЦИАЛИЗАЦИИ ПОРТОВ
DB      0               ;ТЕРМИНАТОР

END
```

10F. ЗАДЕРЖКА В МИЛЛИСЕКУНДАХ (DELAY)

Обеспечивается задержка в диапазоне от 1 до 256 мс, зависящая от заданного параметра. Значение параметра 0 интерпретируется как 256. Пользователь должен вычислить значение CPMS (число циклов в секунду), соответствующее конкретной ЭВМ. Обычно это 2000 для тактовой частоты 2 МГц, 3000 – для 3 МГц и 5000 – для 5 МГц.

Процедура. Регистр B просто уменьшается на величину времени, определяемую константой, подставляемой пользователем. При этом время, необходимое для выполнения команд CALL и RET в программе пользователя и дополнительных команд в подпрограмме остается вообще без изменения.

Используемые регистры: AF, B.
Время выполнения: 1 мс * (A).
Размер программы: 54 байта.
Память, необходимая для данных: отсутствует.
Специальный случай: (A) = 0 вызывает задержку на 256 мс.

УСЛОВИЯ НА ВЫХОДЕ

Время задержки, мс (от 1 до 256), в регистре A.

УСЛОВИЯ НА ВЫХОДЕ

После заданной задержки управление возвращается обратившейся программе, при этом (A) = 0.

1. Данные: (A) = время задержки, мс, = $2A_{16}$.
   Результат: при правильном значении CPMS, подставленном пользователем, программная задержка на $2A_{16}$ мс.

```
;                                                                      ;
;                                                                      ;
;                                                                      ;
;                                                                      ;
;       ЗАГОЛОВОК:      ЗАДЕРЖКА В МИЛЛИСЕКУНДАХ                       ;
;       ИМЯ:            DELAY                                          ;
;                                                                      ;
;                                                                      ;


;                                                                      ;
;       НАЗНАЧЕНИЕ:     ЗАДЕРЖКА ОТ 1 ДО 256 МС                        ;
;                                                                      ;
;       ВХОД:           РЕГИСТР A = ЗАДЕРЖКА, МС. ЗНАЧЕНИЕ 0           ;
;                                   ЭКВИВАЛЕНТНО 256 МС                ;
;                                                                      ;
;       ВЫХОД:          ВОЗВРАТ В ОБРАТИВШУЮСЯ ПРОГРАММУ ПОСЛЕ ЗАДАННОЙ ;
;                       ЗАДЕРЖКИ                                       ;
;                                                                      ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ:  AF,B                                   ;
;                                                                      ;
;       ВРЕМЯ:          1 МС * РЕГИСТР A                               ;
;                                                                      ;
;       РАЗМЕР:         ПРОГРАММА - 54 БАЙТА                           ;
;                                                                      ;
;                                                                      ;


;
;ЭКВИВАЛЕНТНЫЕ ЗНАЧЕНИЯ
;ЧИСЛО ТАКТОВ В МИЛЛИСЕКУНДУ - ЗАДАЕТСЯ ПОЛЬЗОВАТЕЛЕМ
;
CPMS    EQU     2000    ;2000 = ТАКТОВАЯ ЧАСТОТА 2 МГЦ
                        ;3000 = ТАКТОВАЯ ЧАСТОТА 3 МГЦ
                        ;5000 = ТАКТОВАЯ ЧАСТОТА 5 МГЦ
                        ;    .
                        ;    .

;ПРИМЕЧАНИЕ: ВРЕМЕНА ВЫПОЛНЕНИЯ КОМАНД ОПРЕДЕЛЕНЫ В КОММЕНТАРИЯХ.
;            ЕСЛИ ДАНЫ ДВА ЧИСЛА, РАЗДЕЛЕННЫЕ ЗАПЯТОЙ, ТО ПЕРВОЕ
;            ОТНОСИТСЯ К 8080, А ВТОРОЕ - К 8085.
;
;МЕТОД:
; ПОДПРОГРАММА РАЗДЕЛЕНА НА ДВЕ ЧАСТИ. ОБРАЩЕНИЕ К ПОДПРОГРАММЕ "DLY"
; ВЫЗЫВАЕТ ЗАДЕРЖКУ ТОЧНО НА 1 МС МЕНЬШЕ, ЧЕМ ТРЕБУЕТСЯ. В ПОСЛЕДНЕЙ
; ИТЕРАЦИИ ПРИНИМАЕТСЯ ВО ВНИМАНИЕ ВРЕМЯ ОБРАЩЕНИЯ К; "DELAY" И "DLY".
; ЭТО ДОПОЛНИТЕЛЬНОЕ ВРЕМЯ СОСТАВЛЯЕТ:
;               17,18 ТАКТОВ ==> CALL DELAY
;               17,18 ТАКТОВ ==> CALL DLY
;                5, 4 ТАКТА  ==> DCR  A
;               11,12 ТАКТОВ ==> RZ
;                5, 4 ТАКТА  ==> INR  A
;                7, 7 ТАКТОВ ==> MVI  B,(CPMS/100)-1
```

```
;                0,-3 ТАКТА  ==) JNZ  LDLP (ПРЕДНАЗНАЧЕНА ДЛЯ ПОСЛЕДНЕГО
;                                     ПЕРЕХОДА
;                5, 6 ТАКТОВ ==) RNZ
;                5, 6 ТАКТОВ ==) RNZ
;                4, 4 ТАКТА  ==) NOP
;                4, 4 ТАКТА  ==) NOP
;               10,10 ТАКТОВ ==) JMP  DELAY1
;               10,10 ТАКТОВ ==) RET
;               ------
;              100,100 ТАКТОВ ДОПОЛНИТЕЛЬНО

DELAY:
        ;ОСУЩЕСТВИТЬ ЗАДЕРЖКУ НА 1 МС МЕНЬШЕ ЗАДАННОГО ЗНАЧЕНИЯ
                                 ;17,18 ТАКТОВ ДЛЯ ОБРАЩЕНИЯ ПОЛЬЗОВАТЕЛЯ
        CALL    DLY              ;33,34 ТАКТА ДЛЯ ВЫЗОВА И ВОЗВРАЩЕНИЯ
                                 ; В DLY
        INR     A                ;5,4 ТАКТА (ДЕЛАЕТ ФЛАГ Z = 0)
        MVI     B,(CPMS/50)-2    ;7 ТАКТОВ
                                 ;-----
                                 ;62,63

        ;ЧТОБЫ УЧЕСТЬ ДОПОЛНИТЕЛЬНОЕ ВРЕМЯ, ПРИ ЗАДЕРЖКЕ НА
        ; ПОСЛЕДНЮЮ МИЛЛИСЕКУНДУ ВЫПОЛНИТЬ НА 100 ТАКТОВ МЕНЬШЕ
LDLP:
        JMP     LDLY1            ;10 ТАКТОВ
LDLY1:  JMP     LDLY2            ;10 ТАКТОВ
LDLY2:  JMP     LDLY3            ;10 ТАКТОВ
LDLY3:  RZ                       ;5,6 ТАКТОВ (ВОЗВРАТА НИКОГДА НЕ
                                 ; ПРОИСХОДИТ)
        DCR     B                ;5,4 ТАКТА
        JNZ     LDLP             ;10 ТАКТОВ  (10,7 ДЛЯ ПОСЛЕДНЕГО ПРОХОДА)
                                 ;---
                                 ;50 ТАКТОВ
                                 ;(47 ТАКТОВ ДЛЯ ПОСЛЕДНЕГО ПРОХОДА)

        ;ВЫПОЛНИТЬ 38 ТАКТОВ ДЛЯ 8080 И 40 ТАКТОВ ДЛЯ 8085
        ; 2 ДОПОЛНИТЕЛЬНЫХ ТАКТА ДЛЯ 8085 НЕОБХОДИМЫ ДЛЯ УЧЕТА
        ; ВЫПОЛНЕННОЙ РАНЕЕ КОМАНДЫ "JNZ LDLP", ЗАНИМАЮЩЕЙ ПРИ
        ; ПОСЛЕДНЕМ ПРОХОДЕ ТОЛЬКО 7 ТАКТОВ ВМЕСТО 10
        RNZ                      ;5,6 ТАКТОВ
        RNZ                      ;5,6 ТАКТОВ
        NOP                      ;4 ТАКТА
        NOP                      ;4 ТАКТА
        JMP     DELAY1           ;10 ТАКТОВ
DELAY1: RET                      ;10 ТАКТОВ
                                 ;---
                                 ;38,40 ТАКТОВ


        ;***************************
        ;ПОДПРОГРАММА: DLY
        ;НАЗНАЧЕНИЕ:    ОСУЩЕСТВИТЬ ЗАДЕРЖКУ НА 1 МС МЕНЬШЕ ЗАДАННОГО
        ;               ЗНАЧЕНИЯ
        ;ВХОД:  РЕГИСТР А = ПОЛНАЯ ЗАДЕРЖКА, МС
        ;ВЫХОД: ОСУЩЕСТВЛЯЕТСЯ ЗАДЕРЖКА НА 1 МС МЕНЬШЕ ЗАДАННОГО
        ;       ЗНАЧЕНИЯ
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF,B
        ;***************************
```

```
DLY:
        DCR     A               ;5,4 ТАКТА
        RZ                      ;5,6,ТАКТОВ (ЗДЕСЬ ПРОИСХОДИТ ВОЗВРАТ)
        MVI     B,(CPMS/50-1)   ;7 ТАКТОВ
                                ;---
                                ;17 ТАКТОВ

DLP:
        JMP     DLY1            ;10 ТАКТОВ
DLY1:   JMP     DLY2            ;10 ТАКТОВ
DLY2:   JMP     DLY3            ;10 ТАКТОВ
DLY3:   RZ                      ;5,6 ТАКТОВ (ЗДЕСЬ ВОЗВРАТА НИКОГДА НЕ
                                ; ПРОИСХОДИТ)
        DCR     B               ;5,4 ТАКТА

        JNZ     DLP             ;10 ТАКТОВ
                                ;---
                                ;50 ТАКТОВ

        ;ВЫПОЛНИТЬ 33 ТАКТА ДЛЯ 8080 И 36 ТАКТОВ ДЛЯ 8085
        ; 3 ДОПОЛНИТЕЛЬНЫЕ ТАКТА ДЛЯ 8085 НЕОБХОДИМЫ ДЛЯ УЧЕТА
        ; ВЫПОЛНЕННОЙ РАНЕЕ КОМАНДЫ "JNZ LDLP", ЗАНИМАЮЩЕЙ ПРИ ПОСЛЕДНЕМ
        ; ПРОХОДЕ ТОЛЬКО 7 ТАКТОВ ВМЕСТО 10
        RNZ                     ;5,6 ТАКТОВ (ВОЗВРАТА НИКОГДА НЕ
                                ; ПРОИСХОДИТ)
        RNZ                     ;5,6 ТАКТОВ (ВОЗВРАТА НИКОГДА НЕ
                                ; ПРОИСХОДИТ)
        RNZ                     ;5,6 ТАКТОВ (ВОЗВРАТА НИКОГДА НЕ
                                ; ПРОИСХОДИТ)
        NOP                     ;4 ТАКТА
        NOP                     ;4 ТАКТА
        JMP     DLY             ;10 ТАКТОВ


;                                                                    ;
;                                                                    ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                            ;
;                                                                    ;
;                                                                    ;

SC10F:
        ;
        ;ЗАДЕРЖАТЬ НА 10 С
        ; ВЫЗВАТЬ ПОДПРОГРАММУ DELAY 40 РАЗ С ЗАДЕРЖКОЙ В 250 МС
        ;
        MVI     E,40            ;40 РАЗ (ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО 28)
QTRSCD:
        MVI     A,250           ;250 МС (ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО FA)
        CALL    DELAY
        DCR     E               ;УМЕНЬШИТЬ СЧЕТЧИК
        JNZ     QTRSCD          ;ПРОДОЛЖАТЬ, ПОКА НЕ БУДЕТ ВЫПОЛНЕНО

        JMP     SC10F

        END     ;ПРОГРАММА
```

ГЛАВА 11

ПРЕРЫВАНИЯ

11А. НЕБУФЕРИРОВАННЫЙ ВВОД-ВЫВОД ПО ПРЕРЫВАНИЯМ С ИСПОЛЬЗОВАНИЕМ ПРОГРАММИРУЕМОГО ИНТЕРФЕЙСА СВЯЗИ (PCI) 8251 (SINTIO)

Выполняются ввод и вывод по прерываниям с использованием 8251 или USART и односимвольных буферов ввода и вывода. Содержатся следующие подпрограммы:

1) INCH – читает символ из буфера ввода;
2) INST – определяет, пустой ли буфер ввода;
3) OUTCH – записывает символ в буфер вывода;
4) OUTST – определяет, заполнен ли буфер вывода;
5) INIT – инициализирует 8251, векторы прерываний и программируемые флаги. Флаги служат для управления передачей данных между главной программой и подпрограммами обработки прерываний.

Прерывания обслуживаются следующими подпрограммами:

1. RDHDLR отвечает на прерывание по вводу, считывая символ с 8251 в буфер ввода.
2. WRHDLR отвечает на прерывание по выводу, записывая символ из буфера вывода в 8251.

Процедура.

1. INCH – проверяется в цикле, пока символ не станет доступным, очищается флаг готовности данных (RECDF) и загружается символ в аккумулятор.
2. INST – устанавливается флаг переноса равным флагу готовности данных (RECDF).
3. OUTCH – проверяется в цикле, не свободен ли буфер вывода, запоминается символ в буфере и устанавливается флаг готовности символа (TRNDF). Если прерывания не ожидается (т. е. прерывание было очищено, так как оно произошло в момент, когда данные не были готовы), данные немедленно посылаются в интерфейс 8251.
4. OUTST – устанавливается флаг переноса по флагу готовности символа (TRNDF).
5. INIT – очищаются программные флаги, устанавливаются векторы прерываний и инициализируется 8251, при этом соответствующие значения помещаются в регистры способа работы и командный регистр. Для более детального ознакомления с инициализацией 8251 см. подпрограмму 10Е.
6. RDHDLR – считываются данные, сохраняются в буфере ввода и устанавливается флаг их готовности (RECDF).
7. WRHDLR – определяется, доступны ли данные. Если нет, то просто очищается прерывание по выводу. Если данные доступны, то они передаются 8251 и очищается флаг готовности символа (TRNDF).

В большинстве систем прерываний на базе 8080 или 8085 используется контроллер, который отвечает на подтверждения прерываний от центрального процессора и работает с приоритетом, векторами и другими логическими элементами механизма прерываний. В примере, приведенном в листинге, использован популярный программируемый контроллер прерываний (PIC)

8259. Контроллер 8259 фиксирует запросы на прерывания от периферийных устройств, блокирует последующие запросы того же самого и более низких уровней прерывания и генерирует команду CALL, передающую управление вектору прерываний 8080 или 8085. Перед разблокированием последующих запросов подпрограмма, обслуживающая прерывания, должна послать контроллеру 8259 команду конца прерывания (EOI).

Заметим, что когда 8259 работает в обычном режиме "определения фронта", он распознает только передачи по линиям прерываний. Таким образом, даже если прерывание от периферийного устройства и не очищено, оно будет распознано только один раз. Вывод прерывания с устройства будет оставаться активным, однако 8259 не будет генерировать другого прерывания процессора. Устройство 8259 подробно описано в работе Using the 8259 Programmable Interrupt Controller, Intel Application Note AP-31, Intel Corporation, Santa Clara, CA, 1977.

Отдельная проблема, связанная с прерыванием по выводу, состоит в том, что оно может возникнуть в момент, когда нет готовых данных. Его нельзя проигнорировать, так как оно будет возникать непрерывно, создавая, таким образом, бесконечный цикл. Решением является просто очистка прерывания с помощью посылки устройству 8259 команды конца прерывания.

Если бы в системе не было 8259, прерывания по выводу следовало бы запретить. Не существует способа очистить прерывание 8251, не посылая на это устройство данных; это можно сделать для некоторых серийных интерфейсов, таких как Z80.

Однако теперь, когда данные становятся доступными, возникает новая проблема. Она состоит в том, что когда прерывание очищено, совершенно очевидно, система не может быть информирована о готовности устройства к передаче. Решение состоит в том, чтобы завести флаг, который указывает (когда его значение равно 0), что прерывание по выводу уже произошло, но не было обслужено. Этот флаг называется OIE (Output Interrupt Expected — ожидание прерывания по выводу).

Подпрограмма инициализации очищает OIE (так как 8251 обязательно начинает работать с состояния готовности по передаче. Когда происходит прерывание, которое не может быть обслужено (нет доступных данных), подпрограмма, осуществляющая вывод, очищает этот флаг и устанавливает его только после того, как данные посланы интерфейсу 8251 (на тот случай, если флаг мог быть уже очищен). Теперь, чтобы определить, не произошло ли уже прерывания по выводу, подпрограмма OUTCH может проверить OIE (0 указывает на то, что прерывание было, шестнадцатеричное число FF — что не было). Если прерывания по выводу не ожидается, OUTCH просто немедленно посылает данные.

Необслуженные прерывания могут возникнуть только при работе с устройствами вывода, так как в устройствах ввода, когда они обслуживают запрос, всегда есть данные, готовые для передачи. Таким образом, устройства вывода вызывают больше проблем, связанных с инициализацией и последующей работой в системах, управляемых прерываниями, чем устройства ввода.

Используемые регистры:
1. INCH: AF.
2. INST: AF.
3. OUTCH: AF.
4. OUTST: AF.
5. INIT: AF, HL.

Время выполнения:
1. INCH: 105 тактов (8080) или 106 тактов (8085), если символ готов.
2. INST: 27 тактов (8080 или 8085).
3. OUTCH: 198 тактов (8080 или 8085), если буфер вывода пустой и ожидается прерывание по выводу; 74 (8080) или 76 (8085) дополнительных тактов для пересылки данных программируемому интерфейсу, если не ожидается прерывания по выводу.
4. OUTST: 27 тактов (8080 или 8085).
5. INIT: 271 такт (8080 или 8085).
6. RDHDLR: 95 тактов (8080) или 96 тактов (8085).
7. WRHDLR: 174 такта (8080 или 8085), если буфер вывода заполнен; 96 (8080) или 97 (8085) тактов, если буфер вывода пустой.

Размер программы: 166 байт.

Память, необходимая для данных: 5 байт в любом месте ОЗУ для полученных данных (адрес RECDAT), флага полученных данных (адрес RECDF), переданных данных (адрес TRNDAT), флага передачи данных (адрес TRNDF) и флага ожидания прерывания по выводу (адрес OIE).

УСЛОВИЯ НА ВХОДЕ

1. INCH: нет параметров.
2. INST: нет параметров.
3. OUTCH: передаваемый символ в A.
4. OUTST: нет параметров.
5. INIT: нет параметров.

УСЛОВИЯ НА ВЫХОДЕ

1. INCH: символ в регистре A.
2. INST: флаг переноса = 0, если буфер ввода пустой, 1 — если полный.
3. OUTCH: нет параметров.
4. OUTST: флаг переноса = 0, если буфер вывода пустой, 1 — если полный.
5. INIT: нет параметров.

```
;                                                               ;
;                                                               ;
;                                                               ;
;                                                               ;
;       ЗАГОЛОВОК:      НЕБУФЕРИРОВАННЫЙ ВВОД-ВЫВОД С ИСПОЛЬЗОВАНИЕМ ;
;                       ПРОГРАММИРУЕМОГО ИНТЕРФЕЙСА СВЯЗИ 8251  ;
;       ИМЯ:            SINTIO                                  ;
;                                                               ;
;                                                               ;
;                                                               ;
;       НАЗНАЧЕНИЕ:     ЭТА ПРОГРАММА СОДЕРЖИТ 5 ПОДПРОГРАММ, КОТОРЫЕ ;
;                       ВЫПОЛНЯЮТ ВВОД И ВЫВОД ПО ПРЕРЫВАНИЯМ С ;
;                       ИСПОЛЬЗОВАНИЕМ 8251.                    ;
```

```
;                                                                    ;
;                       INCH                                         ;
;                         ПРОЧИТАТЬ СИМВОЛ                           ;
;                       INST                                         ;
;                         ОПРЕДЕЛИТЬ СОСТОЯНИЕ ВВОДА (ПУСТОЙ ЛИ БУФЕР ;
;                         ВВОДА)                                     ;
;                       OUTCH                                        ;
;                         ЗАПИСАТЬ СИМВОЛ                            ;
;                       OUTST                                        ;
;                         ОПРЕДЕЛИТЬ СОСТОЯНИЕ ВЫВОДА (ЗАПОЛНЕН ЛИ БУФЕР;
;                         ВЫВОДА)                                    ;
;                       INIT                                         ;
;                         ИНИЦИАЛИЗИРОВАТЬ ИНТЕРФЕЙС И СИСТЕМУ       ;
;                         ПРЕРЫВАНИЙ                                 ;
;                                                                    ;
;       ВХОД:           INCH                                         ;
;                         НЕТ ПАРАМЕТРОВ                             ;
;                       INST                                         ;
;                         НЕТ ПАРАМЕТРОВ                             ;
;                       OUTCH                                        ;
;                         РЕГИСТР A = ПЕРЕДАВАЕМЫЙ СИМВОЛ            ;
;                       OUTST                                        ;
;                         НЕТ ПАРАМЕТРОВ                             ;
;                       INIT                                         ;
;                         НЕТ ПАРАМЕТРОВ                             ;
;                                                                    ;
;       ВЫХОД:          INCH                                         ;
;                         РЕГИСТР A = СИМВОЛ                         ;
;                       INST                                         ;
;                         ФЛАГ ПЕРЕНОСА = 0, ЕСЛИ БУФЕР ВВОДА ПУСТОЙ, ;
;                         1, ЕСЛИ ЕСТЬ СИМВОЛ                        ;
;                       OUTCH                                        ;
;                         НЕТ ПАРАМЕТРОВ                             ;
;                       OUTST                                        ;
;                         ФЛАГ ПЕРЕНОСА = 0, ЕСЛИ БУФЕР ВЫВОДА ПУСТОЙ, ;
;                         1, ЕСЛИ ОН ПОЛНЫЙ                          ;
;                       INIT                                         ;
;                         НЕТ ПАРАМЕТРОВ                             ;
;                                                                    ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ:                                       ;
;                       INCH                                         ;
;                         AF                                         ;
;                       INST                                         ;
;                         AF                                         ;
;                       OUTCH                                        ;
;                         AF                                         ;
;                       OUTST                                        ;
;                         AF                                         ;
;                       INIT                                         ;
;                         AF,HL                                      ;
;                                                                    ;
;       ВРЕМЯ:          INCH                                         ;
;                         105 ТАКТОВ, ЕСЛИ ЕСТЬ СИМВОЛ, ДЛЯ 8080 И   ;
;                         106 ДЛЯ 8085                               ;
;                       INST                                         ;
;                         27 ТАКТОВ ДЛЯ 8080 И 8085                  ;
;                       OUTCH                                        ;
```

```
;                               198 ТАКТОВ, ЕСЛИ БУФЕР ВЫВОДА НЕ ЗАПОЛНЕН И   ;
;                               ОЖИДАЕТСЯ ПРЕРЫВАНИЕ ПО ВЫВОДУ, ДЛЯ 8080 И    ;
;                               8085                                          ;
;                             OUTST                                           ;
;                               27 ТАКТОВ ДЛЯ 8080 И 8085                     ;
;                             INIT                                            ;
;                               271 ТАКТ ДЛЯ 8080 И 8085                      ;
;                             RDHDLR                                          ;
;                               95 ТАКТОВ ДЛЯ 8080 И 96 ТАКТОВ ДЛЯ 8085       ;
;                             WRHDLR                                          ;
;                               174 ТАКТА ДЛЯ 8080 И 8085, ЕСЛИ БУФЕР ВЫВОДА  ;
;                               ПОЛНЫЙ                                        ;

;                                                                             ;
;       РАЗМЕР:               ПРОГРАММА - 166 БАЙТ                            ;
;                             ДАННЫЕ    -   5 БАЙТ                            ;
;                                                                             ;
;                                                                             ;

        ;ЭКВИВАЛЕНТНЫЕ ЗНАЧЕНИЯ ДЛЯ ПРОГРАММИРУЕМОГО ИНТЕРФЕЙСА СВЯЗИ
        ; (PCI) 8251.
        ; 8251 ПРОГРАММИРУЕТСЯ ДЛЯ СЛЕДУЮЩЕГО РЕЖИМА РАБОТЫ:
        ;  АСИНХРОННЫЕ ОПЕРАЦИИ,
        ;  УМНОЖЕНИЯ СКОРОСТИ НА 16,
        ;  8-РАЗРЯДНЫЕ СИМВОЛЫ,
        ;  2 СТОПОВЫХ РАЗРЯДА.
        ;ПРОИЗВОЛЬНЫЕ АДРЕСА ПОРТОВ 8251
USARTDR EQU     084H            ;РЕГИСТР ДАННЫХ
USARTSR EQU     085H            ;РЕГИСТР СОСТОЯНИЯ
USARTCR EQU     085H            ;РЕГИСТР УПРАВЛЕНИЯ

        ;ВЕКТОРЫ ПРЕРЫВАНИЙ
RDITRP  EQU     0020H           ;ВЕКТОР ПРЕРЫВАНИЯ ДЛЯ ЧТЕНИЯ
WRITRP  EQU     0028H           ;ВЕКТОР ПРЕРЫВАНИЯ ДЛЯ ЗАПИСИ

        ;КОМАНДА НАСТРОЙКИ 8251 НА АСИНХРОННЫЙ РЕЖИМ РАБОТЫ
MODE    EQU     11001110B       ;РАЗРЯДЫ 1,0 = 10 (ПРИЗНАК УМНОЖЕНИЯ
                                ; СКОРОСТИ НА 16)
                                ;РАЗРЯДЫ 3,2 = 11 (8-РАЗРЯДНЫЕ СИМВОЛЫ)
                                ;РАЗРЯД 4 = 0 (БЕЗ РАЗРЯДА ЧЕТНОСТИ)
                                ;РАЗРЯД 5 = 0 (НЕ ИСПОЛЬЗУЕТСЯ)
                                ;РАЗРЯДЫ 7,6 = 11 (2 СТОПОВЫХ РАЗРЯДА)

        ;КОМАНДА АСИНХРОННОЙ РАБОТЫ 8251
CMD     EQU     00010111B       ;РАЗРЯД 0 = 1 (РАЗРЕШИТЬ ПЕРЕДАЧУ)
                                ;РАЗРЯД 1 = 1 (ДАННЫЕ НА ТЕРМИНАЛЕ
                                ; ГОТОВЫ
                                ;РАЗРЯД 2 = 1 (РАЗРЕШИТЬ ПРИЕМ)
                                ;РАЗРЯД 3 = 0 (НЕТ СИМВОЛА BREAK
                                ; (ПРЕРВАТЬ))
                                ;РАЗРЯД 4 = 1 (СБРОСИТЬ ОШИБКУ)
                                ;РАЗРЯД 5 = 0 (РАЗРЕШИТЬ ПЕРЕДАЧУ)
                                ;РАЗРЯД 6 = 0 (НЕТ ЗАПРОСА, КОТОРЫЙ
                                ; ТРЕБУЕТСЯ ПОСЛАТЬ)
                                ;РАЗРЯД 7 = 0 (НЕТ ПРОИЗВОЛЬНОГО
                                ; ПОИСКА)
        ;ЭКВИВАЛЕНТНЫЕ ЗНАЧЕНИЯ ДЛЯ ПРОГРАММИРУЕМОГО КОНТРОЛЛЕРА
        ; ПРЕРЫВАНИЙ (PIC) 8259.
```

```
        ; 8259 ПРОГРАММИРУЕТСЯ ДЛЯ СЛЕДУЮЩЕГО РЕЖИМА РАБОТЫ:
        ;  ОДНО УСТРОЙСТВО (В ОТЛИЧИЕ ОТ НЕСКОЛЬКИХ КОНТРОЛЛЕРОВ 8259),
        ;  РЕЖИМ ПОЛНОСТЬЮ ВЛОЖЕННЫХ ПРЕРЫВАНИЙ,
        ;  ВСЕ ПРЕРЫВАНИЯ РАЗРЕШЕНЫ,
        ;  RESTART 4, АДРЕС 0020, ПРЕРЫВАНИЯ ПО ЧТЕНИЮ,
        ;  RESTART 5, АДРЕС 0028, ПРЕРЫВАНИЯ ПО ЗАПИСИ.
        ;ПРОИЗВОЛЬНЫЕ АДРЕСА ПОРТОВ 8259
PIC0    EQU     90H             ;ПОРТ 1 PIC
PIC1    EQU     91H             ;ПОРТ 2 PIC


        ;КОМАНДНЫЕ СЛОВА ICW1 И ICW2 ДЛЯ ИНИЦИАЛИЗАЦИИ 8259
ICW1    EQU     00010010B       ;РАЗРЯД 0 = 0 (НЕ ИСПОЛЬЗУЕТСЯ)
                                ;РАЗРЯД 1 = 1 (ОДИН КОНТРОЛЛЕР 8259)
                                ;РАЗРЯД 2 = 0 (ИНТЕРВАЛ ВЕКТОРА = 8 БАЙТ)
                                ;РАЗРЯД 3 = 0 (ОПРЕДЕЛЯЕТСЯ ФРОНТ)
                                ;РАЗРЯД 4 = 1 (ФИКСИРОВАННОЕ ЗНАЧЕНИЕ)
                                ;РАЗРЯДЫ 5,6,7 = 0 (РАЗРЯДЫ 5-7 БАЗОВОГО
                                ; АДРЕСА ДЛЯ КОМАНД RST)
ICW2    EQU     0               ;СТАРШИЙ БАЙТ БАЗОВОГО АДРЕСА ДЛЯ КОМАНД
                                ; RST

        ;КОМАНДНОЕ СЛОВО ДЛЯ РАБОТЫ 8259
EOI     EQU     00100000B       ;КОНЕЦ КОМАНДНОГО СЛОВА ПРЕРЫВАНИЯ

        ;ПРОЧИТАТЬ СИМВОЛ
INCH:
        CALL    INST            ;ВЗЯТЬ СОСТОЯНИЕ ВВОДА
        JNC     INCH            ;ЖДАТЬ, ЕСЛИ НЕТ СИМВОЛА
        DI                      ;ЗАПРЕТИТЬ ПРЕРЫВАНИЯ
        SUB     A
        STA     RECDF           ;УКАЗАТЬ, ЧТО БУФЕР ВВОДА ПУСТОЙ
        LDA     RECDAT          ;ВЗЯТЬ СИМВОЛ ИЗ БУФЕРА ВВОДА
        EI                      ;РАЗРЕШИТЬ ПРЕРЫВАНИЯ
        RET

        ;ВОЗВРАТИТЬ СОСТОЯНИЕ ВВОДА (ЕСЛИ ЕСТЬ ДАННЫЕ, ФЛАГ ПЕРЕНОСА = 1)
INST:
        LDA     RECDF           ;ВЗЯТЬ ФЛАГ ГОТОВНОСТИ ДАННЫХ
        RAR                     ;УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА ПО ФЛАГУ
                                ; ГОТОВНОСТИ ДАННЫХ; ФЛАГ ПЕРЕНОСА = 1,
                                ; ЕСЛИ ЕСТЬ СИМВОЛ
        RET

        ;ЗАПИСАТЬ СИМВОЛ
OUTCH:
        PUSH    PSW             ;СОХРАНИТЬ ЗАПИСЫВАЕМЫЙ СИМВОЛ

        ;ЖДАТЬ, ПОКА БУФЕР ВЫВОДА НЕ БУДЕТ ПУСТОЙ, ЗАПОМНИТЬ
        ; СЛЕДУЮЩИЙ СИМВОЛ
WAITOC:
        CALL    OUTST           ;ВЗЯТЬ СОСТОЯНИЕ ВЫВОДА
        JC      WAITOC          ;ЖДАТЬ, ЕСЛИ БУФЕР ВЫВОДА ПОЛНЫЙ
        DI                      ;ЗАПРЕТИТЬ ПРЕРЫВАНИЯ НА ВРЕМЯ
                                ; ПРОВЕРКИ ПРОГРАММНЫХ ФЛАГОВ
        POP     PSW             ;ВЗЯТЬ СИМВОЛ
        STA     TRNDAT          ;ЗАПОМНИТЬ СИМВОЛ В БУФЕРЕ ВЫВОДА
```

```
        MVI     A,0FFH          ;УКАЗАТЬ, ЧТО БУФЕР ВЫВОДА ПОЛНЫЙ
        STA     TRNDF
        LDA     OIE             ;ПРОВЕРИТЬ ФЛАГ ОЖИДАНИЯ ПРЕРЫВАНИЯ
        ORA     A               ; ПО ВЫВОДУ
        CZ      OUTDAT          ;ЕСЛИ ПЕРЫВАНИЕ ПО ВЫВОДУ НЕ ОЖИДАЕТСЯ,
                                ; НЕМЕДЛЕННО ПОСЛАТЬ СИМВОЛ
        EI                      ;ВНОВЬ РАЗРЕШИТЬ ПРЕРЫВАНИЯ
        RET

        ;СОСТОЯНИЕ ВЫВОДА (ФЛАГ ПЕРЕНОСА = 1, ЕСЛИ БУФЕР ВЫВОДА ПОЛНЫЙ)
OUTST:
        LDA     TRNDF           ;ВЗЯТЬ ФЛАГ ПЕРЕДАЧИ
        RAR                     ;УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА ПО ФЛАГУ
                                ; ПЕРЕДАЧИ
        RET                     ; ЕСЛИ БУФЕР ПОЛНЫЙ, ФЛАГ ПЕРЕНОСА = 1

        ;ИНИЦИАЛИЗИРОВАТЬ СИСТЕМУ ПРЕРЫВАНИЙ И 8251
INIT:
        DI                      ;ДЛЯ ИНИЦИАЛИЗАЦИИ ЗАПРЕТИТЬ ПРЕРЫВАНИЯ

        ;ИНИЦИАЛИЗИРОВАТЬ ПРОГРАММНЫЕ ФЛАГИ
        SUB     A
        STA     RECDF           ;НЕТ ДАННЫХ ДЛЯ ВЫВОДА
        STA     TRNDF           ;БУФЕР ВЫВОДА ПУСТОЙ
        STA     OIE             ;УКАЗАТЬ, ЧТО ПРЕРЫВАНИЕ ПО ВЫВОДУ
                                ; НЕ НУЖНО, ТАК КАК 8251 ИЗНАЧАЛЬНО
                                ; ГОТОВ К ПЕРЕДАЧЕ

        ;ИНИЦИАЛИЗИРОВАТЬ ВЕКТОРЫ ПРЕРЫВАНИЙ
        LXI     H,RDHDLR
        MVI     A,0C3H          ;ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО C3 - ЭТО КОД
                                ; ОПЕРАЦИИ КОМАНДЫ JMP
        STA     RDITRP          ;ЗАПОМНИТЬ КОД ОПЕРАЦИИ КОМАНДЫ JMP
                                ; ДЛЯ ПРЕРЫВАНИЙ ПО ЧТЕНИЮ
        SHLD    RDITRP+1        ;ЗАПОМНИТЬ АДРЕС ПЕРЕХОДА
        LXI     H,WRHDLR
        STA     WRITRP          ;ЗАПОМНИТЬ КОД ОПЕРАЦИИ КОМАНДЫ JMP
                                ; ДЛЯ ПРЕРЫВАНИЙ ПО ЗАПИСИ
        SHLD    WRITRP+1        ;ЗАПОМНИТЬ АДРЕС ПЕРЕХОДА

        ;ИНИЦИАЛИЗИРОВАТЬ КОНТРОЛЛЕР ПРЕРЫВАНИЙ 8259
        MVI     A,ICW1          ;ПОСЛАТЬ ПЕРВОЕ СЛОВО В ПОРТ 0 PIC
        OUT     PIC0
        MVI     A,ICW2          ;ПОСЛАТЬ ВТОРОЕ СЛОВО В ПОРТ 1 PIC
        OUT     PIC1

        ;ИНИЦИАЛИЗИРОВАТЬ 8251
        MVI     A,10000000B     ;СБРОСИТЬ ПРОГРАММНО 8251,
        OUT     USARTCR         ; ПОСЛАВ ЕМУ 2 БАЙТА, СОДЕРЖАЩИХ
        OUT     USARTCR         ; ШЕСТНАДЦАТЕРИЧНОЕ ЗНАЧЕНИЕ 80,
        MVI     A,01000000B     ; ЗА КОТОРЫМИ СЛЕДУЕТ КОМАНДА СБРОСА
        OUT     USARTCR

        MVI     A,MODE          ;ВЫДАТЬ БАЙТ РЕЖИМА РАБОТЫ
        OUT     USARTCR
        MVI     A,CMD           ;ВЫДАТЬ КОМАНДНЫЙ БАЙТ
        OUT     USARTCR
        IN      USARTDR         ;ПРОЧИТАТЬ РЕГИСТР ДАННЫХ, ЧТОБЫ
```

```
                                ; ИЗНАЧАЛЬНО ОЧИСТИТЬ РЕГИСТР ВВОДА
                                ; ОТ СЛУЧАЙНЫХ ДАННЫХ

        EI                      ;РАЗРЕШИТЬ ПРЕРЫВАНИЯ
        RET

        ;ДИСПЕТЧЕР ПРЕРЫВАНИЯ ПО ВВОДУ (ЧТЕНИЮ)
RDHDLR:
        PUSH    PSW             ;СОХРАНИТЬ AF
        IN      USARTDR         ;СЧИТАТЬ ДАННЫЕ С 8251
        STA     RECDAT          ;СОХРАНИТЬ ДАННЫЕ В БУФЕРЕ ВВОДА
        MVI     A,0FFH
        STA     RECDF           ;УКАЗАТЬ, ЧТО В БУФЕРЕ ВВОДА ЕСТЬ ДАННЫЕ
        MVI     A,EOI
        OUT     PICO            ;ОЧИСТИТЬ ПРЕРЫВАНИЯ 8259
        POP     PSW             ;ВОССТАНОВИТЬ AF
        EI                      ;ВНОВЬ РАЗРЕШИТЬ ПРЕРЫВАНИЯ
        RET

        ;ДИСПЕТЧЕР ПРЕРЫВАНИЙ ПО ВЫВОДУ (ЗАПИСИ)
WRHDLR:
        PUSH    PSW             ;СОХРАНИТЬ AF
        LDA     TRNDF           ;ПРОВЕРИТЬ ФЛАГ НАЛИЧИЯ ДАННЫХ
        ORA     A
        JZ      NODATA          ;ПЕРЕЙТИ, ЕСЛИ НЕТ ДАННЫХ ДЛЯ ПЕРЕДАЧИ
        CALL    OUTDAT          ;ПОСЛАТЬ ДАННЫЕ В 8251
        JMP     WRDONE

        ;ЕСЛИ ПРОИЗОШЛО ПРЕРЫВАНИЕ ПРИ ОТСУТСТВИИ ДАННЫХ, МЫ ДОЛЖНЫ ЕГО
        ; ОЧИСТИТЬ (В 8259), ЧТОБЫ ИЗБЕЖАТЬ ЗАЦИКЛИВАНИЯ. ПОЗЖЕ, КОГДА
        ; СИМВОЛ СТАНОВИТСЯ ДОСТУПНЫМ, НЕОБХОДИМО ЗНАТЬ, ЧТО БЫЛО
        ; ПРЕРЫВАНИЕ ПО ВЫВОДУ, КОТОРОЕ ЕЩЕ НЕ ОБСЛУЖИВАЛОСЬ. ЭТО МОЖНО
        ; ОПРЕДЕЛИТЬ ПО ФЛАГУ OIE ОЖИДАНИЯ ПРЕРЫВАНИЯ ПО ВЫВОДУ. КОГДА
        ; ПРЕРЫВАНИЕ ПРОИСХОДИТ, НО ОСТАЕТСЯ НЕОБСЛУЖЕННЫМ, ЭТОТ ФЛАГ
        ; ОЧИЩАЕТСЯ. КРОМЕ ТОГО, ОН ОЧИЩАЕТСЯ С САМОГО НАЧАЛА, ТАК КАК
        ; 8251 НАЧИНАЕТ РАБОТАТЬ С ГОТОВНОСТЬЮ ПО ВЫВОДУ. OIE
        ; УСТАНАВЛИВАЕТСЯ КАЖДЫЙ РАЗ, КОГДА ДАННЫЕ В ДЕЙСТВИТЕЛЬНОСТИ
        ; ПОСЫЛАЮТСЯ НА 8251. ТАКИМ ОБРАЗОМ, ПОДПРОГРАММА OUTCH МОЖЕТ
        ; ПРОВЕРИТЬ OIE, ЧТОБЫ ОПРЕДЕЛИТЬ, СЛЕДУЕТ ЛИ ПОСЫЛАТЬ ДАННЫЕ
        ; НЕМЕДЛЕННО ИЛИ ЖДАТЬ ПРЕРЫВАНИЯ ПО ВЫВОДУ.
        ;ПРОБЛЕМА ЗДЕСЬ СОСТОИТ В ТОМ, ЧТО УСТРОЙСТВО ВЫВОДА МОЖЕТ
        ; ЗАПРОСИТЬ ОБСЛУЖИВАНИЕ ДО ТОГО, КАК У ЭВМ ПОЯВИТСЯ ЧТО-ЛИБО
        ; ДЛЯ ПЕРЕДАЧИ УСТРОЙСТВУ (В ОТЛИЧИЕ ОТ УСТРОЙСТВА ВВОДА,
        ; КОТОРОЕ, ЗАПРАШИВАЯ ОБСЛУЖИВАНИЕ, ИМЕЕТ ДАННЫЕ). ЭТА ПРОБЛЕМА
        ; НЕОБСЛУЖЕННЫХ ПРЕРЫВАНИЯ ПО ВЫВОДУ РЕШАЕТСЯ С ПОМОЩЬЮ
        ; МНОГОКРАТНОЙ УСТАНОВКИ ФЛАГА OIE, ПОЗВОЛЯЮЩЕГО РАСПОЗНАВАТЬ
        ; ЭТИ ПРЕРЫВАНИЯ.
NODATA:
        SUB     A
        STA     OIE             ;ПРЕРЫВАНИЯ НЕ ОЖИДАЕТСЯ

WRDONE:
        MVI     A,EOI           ;ОЧИСТИТЬ ПРЕРЫВАНИЕ НА 8259
        OUT     PICO
        POP     PSW             ;ВОССТАНОВИТЬ AF
        EI                      ;ВНОВЬ РАЗРЕШИТЬ ПРЕРЫВАНИЯ
        RET
```

```
        ;*********************************************
        ;ПОДПРОГРАММА: OUTDAT
        ;НАЗНАЧЕНИЕ:   ПОСЫЛАЕТ USART СИМВОЛ
        ;ВХОД:  TRNDAT = СИМВОЛ
        ;ВЫХОД: НЕТ ПАРАМЕТРОВ
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF
        ;*********************************************
OUTDAT:
        LDA     TRNDAT          ;ВЗЯТЬ ДАННЫЕ ИЗ БУФЕРА ВЫВОДА
        OUT     USARTDR         ;ПОСЛАТЬ ДАННЫЕ 8251
        SUB     A               ;ОТМЕТИТЬ, ЧТО БУФЕР ВЫВОДА ПУСТОЙ
        STA     TRNDF
        DCR     A               ;ОТМЕТИТЬ, ЧТО ОЖИДАЕТСЯ ПРЕРЫВАНИЕ ПО
        STA     OIE             ; ВЫВОДУ, OIE = ШЕСТНАДЦАТЕРИЧНОЕ
                                ; ЧИСЛО FF
        RET

        ;СЕКЦИЯ ДАННЫХ
RECDAT: DS      1               ;ПОЛУЧАЕМЫЕ ДАННЫЕ
RECDF:  DS      1               ;ФЛАГ ПОЛУЧАЕМЫХ ДАННЫХ
                                ; (0 = НЕТ ДАННЫХ, FF = ДАННЫЕ ЕСТЬ)
TRNDAT: DS      1               ;ПЕРЕДАВАЕМЫЕ ДАННЫЕ
TRNDF:  DS      1               ;ФЛАГ ПЕРЕДАВАЕМЫХ ДАННЫХ
                                ; (0 = БУФЕР ПУСТОЙ, FF = БУФЕР ПОЛНЫЙ)
OIE:    DS      1               ;ФЛАГ ОЖИДАНИЯ ПРЕРЫВАНИЯ ПО ВЫВОДУ
                                ; (0 = ПРЕРЫВАНИЯ НЕ ОЖИДАЕТСЯ,
                                ; FF = ПРЕРЫВАНИЕ ОЖИДАЕТСЯ)


;                                                                       ;
;                                                                       ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                               ;
;                                                                       ;
;                                                                       ;

        ;ЭКВИВАЛЕНТНЫЕ СИМВОЛЫ
ESCAPE  EQU     1BH             ;СИМВОЛ ASCII ESCAPE
TESTCH  EQU     'A'             ;СИМВОЛ ДЛЯ ТЕСТА = A

SC11A:
        CALL    INIT            ;ИНИЦИАЛИЗИРОВАТЬ 8251, СИСТЕМУ
                                ; ПРЕРЫВАНИЯ

        ;ПРОСТОЙ ПРИМЕР - ЧИТАТЬ И ВЫДАВАТЬ СИМВОЛЫ, ПОКА НЕ БУДЕТ
        ; ПОЛУЧЕН СИМВОЛ ESC
LOOP:
        CALL    INCH            ;ПРОЧИТАТЬ СИМВОЛ
        PUSH    PSW
        CALL    OUTCH           ;ВЫДАТЬ СИМВОЛ
        POP     PSW
        CPI     ESCAPE          ;СИМВОЛ ESCAPE?
        JNZ     LOOP            ;ЕСЛИ НЕТ, ОСТАТЬСЯ В ЦИКЛЕ

        ;ПРИМЕР АСИНХРОННОЙ РАБОТЫ
        ; ВЫДАВАТЬ НА КОНСОЛЬ "A", НО ОДНОВРЕМЕННО СЛЕДИТЬ И ЗА
        ; СТОРОНОЙ ВХОДА, ЧИТАЯ И ВЫДАВАЯ ЛЮБЫЕ ВВОДИМЫЕ СИМВОЛЫ
```

```
ASYNLP:
        ;ЕСЛИ ВЫВОД НЕ ЗАНЯТ, ВЫВОДИТЬ "A"
        CALL    OUTST               ;ВЫВОД ЗАНЯТ?
        JC      ASYNLP              ;ЕСЛИ ДА, ПЕРЕЙТИ
        MVI     A,TESTCH
        CALL    OUTCH               ;ВЫВЕСТИ ТЕСТОВЫЙ СИМВОЛ

        ;ПРОВЕРИТЬ ПОРТ ВВОДА
        ;ВЫВЕСТИ СИМВОЛ, ЕСЛИ ОН ЕСТЬ
        ;ВЫЙТИ ПО СИМВОЛУ ESCAPE
        CALL    INST                ;ЕСТЬ ДАННЫЕ НА ВХОДЕ?
        JC      ASYNLP              ;ЕСЛИ НЕТ, ПЕРЕЙТИ (ПОСЛАТЬ ЕЩЕ ОДИН
                                    ; СИМВОЛ "A")
        CALL    INCH                ;ВЗЯТЬ СИМВОЛ
        CPI     ESCAPE              ;ЭТО ESCAPE?
        JZ      DONE                ;ЕСЛИ ДА, ПЕРЕЙТИ
        CALL    OUTCH               ;ИНАЧЕ ВЫДАТЬ СИМВОЛ
        JMP     ASYNLP              ;И ПРОДОЛЖИТЬ РАБОТУ

DONE:
        JMP     SC11A

        END
```

11В. НЕБУФЕРИРОВАННЫЙ ВВОД-ВЫВОД С ИСПОЛЬЗОВАНИЕМ ПРОГРАММИРУЕМОГО ПЕРИФЕРИЙНОГО ИНТЕРФЕЙСА (PPI) 8255 (PINTIO)

Выполняются ввод и вывод по прерываниям с использованием 8255 и односимвольных буферов ввода и вывода. Содержатся следующие подпрограммы:

1) INCH – читает символ из буфера ввода;

2) INST – определяет, пустой ли буфер вывода;

3) OUTCH – записывает символ в буфер вывода;

4) OUTST – определяет, заполнен ли буфер вывода;

5) INIT – инициализирует 8255, векторы прерываний и программные флаги. Флаги используются для управления передачей данных между главной программой и подпрограммами обслуживания прерываний.

Прерывания обслуживаются следующими подпрограммами:

1) RDHDLR – отвечает на прерывание по вводу, считывая символ с 8255 в буфер ввода;

2) WRHDLR – отвечает на прерывание по выводу, записывая символ из буфера вывода в 8255.

Процедура.

1. INCH – проверяется в цикле, не стал ли символ доступным, очищается флаг готовности данных (RECDF) и загружается символ в аккумулятор.

2. INST – устанавливается флаг переноса равным флагу готовности данных (RECDF).

3. OUTCH – проверяется в цикле, свободен ли буфер вывода, запоминается символ в буфере и устанавливается флаг готовности символа (TRNDF). Если прерывания не ожидается (т. е. прерывание было очищено, так как оно произошло в момент, когда данные не были готовы), данные немедленно посылаются 8255.

4. OUTST – устанавливается флаг переноса по флагу готовности символа (TRNDF).

5. INIT – очищаются программные флаги, устанавливаются векторы прерываний и инициализируется 8255 с загрузкой его регистров управления и регистра вектора прерываний. Для более детального ознакомления с инициализацией 8255 см. гл. 1 и подпрограмму 10E.

6. RDHDLR – символ считывается с 8255, сохраняется в буфере ввода и устанавливается флаг готовности данных (RECDF).

7. WRHDLR – определяется, доступны ли данные для вывода. Если нет, просто очищается прерывание по выводу. Если данные доступны, то передаются устройству 8255 и очищается флаг готовности символа (TRNDF).

В большинстве систем прерываний на базе микропроцессора 8080 или 8085 используется контроллер, который отвечает на подтверждения прерываний от центрального процессора и работает с приоритетом, векторами и другими логическими элементами механизма прерываний. В примере, приведенном в листинге, использован популярный программируемый контроллер прерываний 8259. Этот контроллер фиксирует запросы на прерывания от периферийных устройств, блокирует последующие запросы того же самого или более низких уровней прерываний и генерирует команду CALL, передающую управление вектору прерываний 8080 или 8085. Перед разблокированием последующих запросов подпрограмма, обслуживающая прерывания, должна послать контроллеру 8259 команду конца прерывания (EOI).

Заметим, что когда 8259 работает в обычном режиме "определения фронта", он распознает только передачи по линиям прерываний. Таким образом, даже если прерывание от периферийного устройства и не очищено, оно будет распознано только один раз. Вывод прерывания с устройства будет оставаться активным, однако 8259 не будет генерировать другого прерывания процессора. Устройство 8259 подробно описано в работе, упомянутой на стр. 363.

Если бы в системе не было 8259, прерывания по выводу следовало бы запретить. Не существует способа очистить прерывание 8255 прямо, не посылая на это устройство данных; это можно сделать для некоторых параллельных интерфейсов, таких как адаптер периферийного интерфейса 6820 и универсальный интерфейсный адаптер 6522.

Отдельная проблема, связанная с прерыванием по выводу, может возникнуть, когда нет готовых данных, для того чтобы их можно было послать. Это прерывание нельзя проигнорировать, так как оно будет непрерывно возникать, создавая, таким образом, бесконечный цикл. Решением является просто очистка прерывания с помощью посылки устройству 8259 команды конца прерывания.

Однако теперь, когда данные становятся доступными, возникает новая проблема. Она состоит в том, что когда прерывание очищено, оно, совершенно очевидно, не может информировать систему о том, что устройство вывода готово прочитать данные. Решение состоит в том, чтобы завести флаг, который указывает (когда его значение равно 0), что прерывание по выводу уже произошло, но не было обслужено. Этот флаг называется OIE – ожидание прерываний по выводу.

Подпрограмма инициализации очищает OIE (так как устройство вывода начинает работать в состоянии готовности к передаче). Когда происходит

прерывание, которое не может быть обслужено (нет доступных данных), подпрограмма, осуществляющая вывод, очищает этот флаг и устанавливает его только после того, как данные посланы интерфейсу 8255 (на тот случай, если флаг мог быть уже очищен). Теперь, чтобы определить, не произошло ли уже прерывания по выводу, подпрограмма OUTCH может проверить OIE. Если прерывания по выводу не ожидается, подпрограмма OUTCH просто посылает немедленно данные.

Необслуженные прерывания могут возникнуть только при работе с устройствами вывода, так как устройства ввода, когда они обслуживают запрос, всегда имеют данные, готовые для передачи. Таким образом, в системах, работающих по прерываниям, устройства вывода вызывают больше проблем, связанных с инициализацией и последующей работой, чем устройства ввода.

Используемые регистры:
1. INCH: AF.
2. INST: AF.
3. OUTCH: AF.
4. OUTST: AF.
5. INIT: AF, HL.

Время выполнения:
1. INCH: 102 такта (8080) или 100 тактов (8085), если символ готов.
2. INST: 27 тактов (8080 или 8085).
3. OUTCH: 143 такта (8080) или 139 тактов (8085), если буфер вывода не полный и ожидается прерывание по выводу; 74 (8080) или 76 (8085) дополнительных тактов для пересылки данных интерфейсу 8255, если не ожидается прерывания по выводу.
4. OUTST: 27 тактов (8080 или 8085).
5. INIT: 231 такт (8080 или 8085).
6. RDHDLR: 95 тактов (8080) или 96 тактов (8085).
7. WRHDLR: 171 такт (8080) или 169 тактов (8085), если буфер вывода заполнен; 96 тактов (8080) или 97 тактов (8085), если буфер вывода пустой.

Размер программы: 162 байта.

Память, необходимая для данных: 5 байт в любом месте ОЗУ для полученных данных (адрес RECDAT), флага полученных данных (адрес RECDF), передаваемых данных (адрес TRNDAT), флага передачи данных (адрес TRNDF) и флага ожидания прерывания по выводу (адрес OIE).

УСЛОВИЯ НА ВХОДЕ

1. INCH: нет параметров.
2. INST: нет параметров.
3. OUTCH: передаваемый символ в регистре A.
4. OUTST: нет параметров.
5. INIT: нет параметров.

УСЛОВИЯ НА ВЫХОДЕ

1. INCH: символ в регистре A.
2. INST: флаг переноса = 0, если буфер ввода пустой, 1 — если полный.
3. OUTCH: нет параметров.
4. OUTST: флаг переноса = 0, если буфер вывода пустой, 1 — если полный.
5. INIT: нет параметров.

```
;                                                                      ;
;                                                                      ;
;                                                                      ;
;                                                                      ;
;       ЗАГОЛОВОК:      НЕБУФЕРИРОВАННЫЙ ВВОД-ВЫВОД С ИСПОЛЬЗОВАНИЕМ   ;
;                       ПРОГРАММИРУЕМОГО ПАРАЛЛЕЛЬНОГО ИНТЕРФЕЙСА 8255 ;
;       ИМЯ:            PINTIO                                         ;
;                                                                      ;
;                                                                      ;


;                                                                      ;
;       НАЗНАЧЕНИЕ:     ЭТА ПРОГРАММА СОДЕРЖИТ 5 ПОДПРОГРАММ, КОТОРЫЕ  ;
;                       ВЫПОЛНЯЮТ ВВОД И ВЫВОД ПО ПРЕРЫВАНИЯМ С        ;
;                       ИСПОЛЬЗОВАНИЕМ 8255.                           ;
;                                                                      ;
;                       INCH                                           ;
;                         ПРОЧИТАТЬ СИМВОЛ                             ;
;                       INST                                           ;
;                         ОПРЕДЕЛИТЬ СОСТОЯНИЕ ВВОДА (ПУСТОЙ ЛИ БУФЕР  ;
;                         ВВОДА)                                       ;
;                       OUTCH                                          ;
;                         ЗАПИСАТЬ СИМВОЛ                              ;
;                       OUTST                                          ;
;                         ОПРЕДЕЛИТЬ СОСТОЯНИЕ ВЫВОДА (ЗАПОЛНЕН ЛИ БУФЕР;
;                         ВЫВОДА)                                      ;
;                       INIT                                           ;
;                         ИНИЦИАЛИЗИРОВАТЬ 8255 И СИСТЕМУ ПРЕРЫВАНИЙ   ;
;                                                                      ;
;       ВХОД:           INCH                                           ;
;                         НЕТ ПАРАМЕТРОВ                               ;
;                       INST                                           ;
;                         НЕТ ПАРАМЕТРОВ                               ;
;                       OUTCH                                          ;
;                         РЕГИСТР A = ПЕРЕДАВАЕМЫЙ СИМВОЛ              ;
;                       OUTST                                          ;
;                         НЕТ ПАРАМЕТРОВ                               ;
;                       INIT                                           ;
;                         НЕТ ПАРАМЕТРОВ                               ;
;                                                                      ;
;       ВЫХОД:          INCH                                           ;
;                         РЕГИСТР A = СИМВОЛ                           ;
;                       INST                                           ;
;                         ФЛАГ ПЕРЕНОСА = 0, ЕСЛИ БУФЕР ВВОДА ПУСТОЙ,  ;
;                         1, ЕСЛИ ЕСТЬ СИМВОЛ                          ;
;                       OUTCH                                          ;
;                         НЕТ ПАРАМЕТРОВ                               ;
;                       OUTST                                          ;
;                         ФЛАГ ПЕРЕНОСА = 0, ЕСЛИ БУФЕР ВЫВОДА ПУСТОЙ, ;
;                         1, ЕСЛИ ОН ПОЛНЫЙ                            ;
;                       INIT                                           ;
;                         НЕТ ПАРАМЕТРОВ                               ;
;                                                                      ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ:                                         ;
;                       INCH                                           ;
;                         AF                                           ;
;                       INST                                           ;
;                         AF                                           ;
;                       OUTCH                                          ;
```

```
;                          AF                                          ;
;                         OUTST                                        ;
;                          AF                                          ;
;                         INIT                                         ;
;                          AF,HL                                       ;
;                                                                      ;
;       ВРЕМЯ:            INCH                                         ;
;                          102 ТАКТА, ЕСЛИ ЕСТЬ СИМВОЛ, ДЛЯ 8080 И     ;
;                          100 ДЛЯ 8085                                ;
;                         INST                                         ;
;                          27 ТАКТОВ ДЛЯ 8080 И 8085                   ;
;                         OUTCH                                        ;
;                          143 ТАКТА, ЕСЛИ БУФЕР ВЫВОДА НЕ ЗАПОЛНЕН И  ;
;                          ОЖИДАЕТСЯ ПРЕРЫВАНИЕ ПО ВЫВОДУ, ДЛЯ 8080 И  ;
;                          139 ДЛЯ 8085                                ;
;                         OUTST                                        ;
;                          27 ТАКТОВ ДЛЯ 8080 И 8085                   ;
;                         INIT                                         ;
;                          231 ТАКТ ДЛЯ 8080 И 8085                    ;
;                         RDHDLR                                       ;
;                          95 ТАКТОВ ДЛЯ 8080 И 96 ТАКТОВ ДЛЯ 8085     ;
;                         WRHDLR                                       ;
;                          171 ТАКТ ДЛЯ 8080 И 169 ДЛЯ 8085, ЕСЛИ БУФЕР ;
;                          ВЫВОДА ПОЛНЫЙ                               ;

;                                                                      ;
;       РАЗМЕР:           ПРОГРАММА - 162 БАЙТА                        ;
;                         ДАННЫЕ    -   5 БАЙТ                         ;
;                                                                      ;
;                                                                      ;

        ;ЭКВИВАЛЕНТНЫЕ ЗНАЧЕНИЯ ДЛЯ ПРОГРАММИРУЕМОГО ПАРАЛЛЕЛЬНОГО
        ; ИНТЕРФЕЙСА (PPI) 8255
        ; 8255 ПРОГРАММИРУЕТСЯ ДЛЯ СЛЕДУЮЩЕГО РЕЖИМА РАБОТЫ:
        ;  РЕЖИМ РАБОТЫ 1 (СТРОБИРУЕМЫЕ ВВОД И ВЫВОД),
        ;  ПОРТ A - ВВОД,
        ;  ПОРТ B - ВЫВОД,
        ;  ПОРТ C - СИГНАЛЫ ПОДТВЕРЖДЕНИЯ.
        ;ПРОИЗВОЛЬНЫЕ АДРЕСА ПОРТОВ 8255
PPIA    EQU     90H              ;ПОРТ A ДАННЫХ
PPIB    EQU     91H              ;ПОРТ B ДАННЫХ
PPIC    EQU     92H              ;ПОРТ C ДАННЫХ
PPICTRL EQU     93H              ;ПОРТ УПРАВЛЕНИЯ

        ;УПРАВЛЯЮЩИЕ БАЙТЫ 8255
CTRLWRD EQU     010110100B       ;УПРАВЛЯЮЩЕЕ СЛОВО ДЛЯ РЕЖИМА 1,
                                 ; ПОРТ A - ВВОД, ПОРТ B - ВЫВОД
PAIE    EQU     00001001B        ;РАЗРЕШИТЬ ПРЕРЫВАНИЯ ОТ ПОРТА A -
                                 ; УСТАНОВИТЬ РАЗРЯД 4 ПОРТА C
PAID    EQU     00001000B        ;ЗАПРЕТИТЬ ПРЕРЫВАНИЯ ОТ ПОРТА A -
                                 ; ОЧИСТИТЬ РАЗРЯД 4 ПОРТА C
PBIE    EQU     00000101B        ;РАЗРЕШИТЬ ПРЕРЫВАНИЯ ОТ ПОРТА B -
                                 ; УСТАНОВИТЬ РАЗРЯД 2 ПОРТА C
PBID    EQU     00000100B        ;ЗАПРЕТИТЬ ПРЕРЫВАНИЯ ОТ ПОРТА B -
                                 ; ОЧИСТИТЬ РАЗРЯД 2 ПОРТА C

        ;ЭКВИВАЛЕНТНЫЕ ЗНАЧЕНИЯ ДЛЯ ПРОГРАММИРУЕМОГО КОНТРОЛЛЕРА
        ; ПРЕРЫВАНИЙ (PIC) 8259.
```

```
        ; 8259 ПРОГРАММИРУЕТСЯ ДЛЯ СЛЕДУЮЩЕГО РЕЖИМА РАБОТЫ:
        ;  ОДНО УСТРОЙСТВО (В ОТЛИЧИЕ ОТ НЕСКОЛЬКИХ КОНТРОЛЛЕРОВ 8259),
        ;  РЕЖИМ ПОЛНОСТЬЮ ВЛОЖЕННЫХ ПРЕРЫВАНИЙ,
        ;  ВСЕ ПРЕРЫВАНИЯ РАЗРЕШЕНЫ,
        ;  RESTART 4, АДРЕС 0020, ПРЕРЫВАНИЯ ПО ЧТЕНИЮ,
        ;  RESTART 5, АДРЕС 0028, ПРЕРЫВАНИЯ ПО ЗАПИСИ.
        ;АДРЕСА ПОРТОВ 8259
PIC0    EQU     0A0H            ;ПОРТ 1 PIC
PIC1    EQU     0A1H            ;ПОРТ 2 PIC

        ;ВЕКТОРЫ ПРЕРЫВАНИЙ
RDITRP  EQU     0020H           ;ВЕКТОР ПРЕРЫВАНИЯ ДЛЯ ЧТЕНИЯ
WRITRP  EQU     0028H           ;ВЕКТОР ПРЕРЫВАНИЯ ДЛЯ ЗАПИСИ


        ;КОМАНДНЫЕ СЛОВА ICW1 И ICW2 ДЛЯ ИНИЦИАЛИЗАЦИИ 8259
ICW1    EQU     00010010B       ;РАЗРЯД 0 = 0 (НЕ ИСПОЛЬЗУЕТСЯ)
                                ;РАЗРЯД 1 = 1 (ОДИН КОНТРОЛЛЕР 8259)
                                ;РАЗРЯД 2 = 0 (ИНТЕРВАЛ ВЕКТОРА = 8 БАЙТ)
                                ;РАЗРЯД 3 = 0 (ПРЕРЫВАНИЯ
                                ; СИНХРОНИЗИРОВАНЫ ПО ГРАНИЦЕ)
                                ;РАЗРЯД 4 = 1 (ФИКСИРОВАННОЕ ЗНАЧЕНИЕ)
                                ;РАЗРЯДЫ 7,6,5 = 0 (РАЗРЯДЫ 5-7 БАЗОВОГО
                                ; АДРЕСА ДЛЯ КОМАНД RST)
ICW2    EQU     0               ;СТАРШИЙ БАЙТ БАЗОВОГО АДРЕСА ДЛЯ КОМАНД
                                ; RST

        ;КОМАНДНОЕ СЛОВО ДЛЯ РАБОТЫ 8259
EOI     EQU     00100000B       ;КОНЕЦ КОМАНДНОГО СЛОВА ПРЕРЫВАНИЯ

        ;ПРОЧИТАТЬ СИМВОЛ
INCH:
        CALL    INST            ;ВЗЯТЬ СОСТОЯНИЕ ВВОДА
        JNC     INCH            ;ЖДАТЬ, ЕСЛИ НЕТ СИМВОЛА
        DI                      ;ЗАПРЕТИТЬ ПРЕРЫВАНИЯ
        SUB     A
        STA     RECDF           ;УКАЗАТЬ, ЧТО БУФЕР ВВОДА ПУСТОЙ
        LDA     RECDAT          ;ВЗЯТЬ СИМВОЛ ИЗ БУФЕРА ВВОДА
        EI                      ;РАЗРЕШИТЬ ПРЕРЫВАНИЯ
        RET

        ;ВОЗВРАТИТЬ СОСТОЯНИЕ ВВОДА (ЕСЛИ ЕСТЬ ДАННЫЕ, ФЛАГ ПЕРЕНОСА = 1)
INST:
        LDA     RECDF           ;ВЗЯТЬ ФЛАГ ГОТОВНОСТИ ДАННЫХ
        RAR                     ;УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА ПО ФЛАГУ
                                ; ГОТОВНОСТИ ДАННЫХ; ФЛАГ ПЕРЕНОСА = 1,
                                ; ЕСЛИ ЕСТЬ СИМВОЛ
        RET

        ;ЗАПИСАТЬ СИМВОЛ
OUTCH:
        PUSH    PSW             ;СОХРАНИТЬ ЗАПИСЫВАЕМЫЙ СИМВОЛ

        ;ЖДАТЬ, ПОКА БУФЕР ВЫВОДА НЕ БУДЕТ ПУСТОЙ, ЗАПОМНИТЬ
        ; СЛЕДУЮЩИЙ СИМВОЛ
WAITOC:
        CALL    OUTST           ;ВЗЯТЬ СОСТОЯНИЕ ВЫВОДА
```

```
        JC      WAITOC              ;ЖДАТЬ, ЕСЛИ БУФЕР ВЫВОДА ПОЛНЫЙ
        DI                          ;ЗАПРЕТИТЬ ПРЕРЫВАНИЯ НА ВРЕМЯ
                                    ; ПРОВЕРКИ ПРОГРАММНЫХ ФЛАГОВ
        POP     PSW                 ;ВЗЯТЬ СИМВОЛ
        STA     TRNDAT              ;ЗАПОМНИТЬ СИМВОЛ В БУФЕРЕ ВЫВОДА
        MVI     A,0FFH              ;УКАЗАТЬ, ЧТО БУФЕР ВЫВОДА ПОЛНЫЙ
        STA     TRNDF
        LDA     OIE                 ;ПРОВЕРИТЬ ФЛАГ ОЖИДАНИЯ ПРЕРЫВАНИЯ
        ORA     A                   ; ПО ВЫВОДУ
        CZ      OUTDAT              ;ЕСЛИ ПРЕРЫВАНИЯ ПО ВЫВОДУ НЕ ОЖИДАЕТСЯ,
                                    ; НЕМЕДЛЕННО ПОСЛАТЬ СИМВОЛ
        EI                          ;ВНОВЬ РАЗРЕШИТЬ ПРЕРЫВАНИЯ
        RET

        ;СОСТОЯНИЕ ВЫВОДА (ФЛАГ ПЕРЕНОСА = 1, ЕСЛИ БУФЕР ВЫВОДА ПОЛНЫЙ)
OUTST:
        LDA     TRNDF               ;ВЗЯТЬ ФЛАГ ПЕРЕДАЧИ
        RAR                         ;УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА ПО ФЛАГУ
                                    ; ПЕРЕДАЧИ
        RET                         ; ЕСЛИ БУФЕР ПОЛНЫЙ, ФЛАГ ПЕРЕНОСА = 1

        ;ИНИЦИАЛИЗИРОВАТЬ СИСТЕМУ ПРЕРЫВАНИЙ И 8255
INIT:
        DI                          ;ДЛЯ ИНИЦИАЛИЗАЦИИ ЗАПРЕТИТЬ ПРЕРЫВАНИЯ

        ;ИНИЦИАЛИЗИРОВАТЬ ПРОГРАММНЫЕ ФЛАГИ
        SUB     A
        STA     RECDF               ;НЕТ ДАННЫХ ДЛЯ ВВОДА
        STA     TRNDF               ;БУФЕР ВЫВОДА ПУСТОЙ
        STA     OIE                 ;УКАЗАТЬ, ЧТО ПРЕРЫВАНИЕ ПО ВЫВОДУ
                                    ; НЕ НУЖНО, ТАК КАК 8255 ИЗНАЧАЛЬНО
                                    ; ГОТОВ К ПЕРЕДАЧЕ

        ;ИНИЦИАЛИЗИРОВАТЬ ВЕКТОРЫ ПРЕРЫВАНИЯ
        LXI     H,RDHDLR
        MVI     A,0C3H              ;ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО C3 - ЭТО КОД
                                    ; ОПЕРАЦИИ КОМАНДЫ JMP
        STA     RDITRP              ;ЗАПОМНИТЬ КОД ОПЕРАЦИИ КОМАНДЫ JMP
                                    ; ДЛЯ ПРЕРЫВАНИЙ ПО ЧТЕНИЮ
        SHLD    RDITRP+1            ;ЗАПОМНИТЬ АДРЕС ПЕРЕХОДА
        LXI     H,WRHDLR
        STA     WRITRP              ;ЗАПОМНИТЬ КОД ОПЕРАЦИИ КОМАНДЫ JMP
                                    ; ДЛЯ ПРЕРЫВАНИЯ ПО ЗАПИСИ
        SHLD    WRITRP+1            ;ЗАПОМНИТЬ АДРЕС ПЕРЕХОДА

        ;ИНИЦИАЛИЗИРОВАТЬ КОНТРОЛЛЕР ПРЕРЫВАНИЯ 8259
        MVI     A,ICW1              ;ПОСЛАТЬ ПЕРВОЕ СЛОВО В ПОРТ 0 PIC
        OUT     PIC0
        MVI     A,ICW2              ;ПОСЛАТЬ ВТОРОЕ СЛОВО В ПОРТ 1 PIC
        OUT     PIC1

        ;ИНИЦИАЛИЗИРОВАТЬ 8255
        MVI     A,CTRLWRD           ;РЕЖИМ РАБОТЫ 1, ПОРТ A - ВВОД,
                                    ; ПОРТ B - ВЫВОД
        OUT     PPICTRL             ;ИНИЦИАЛИЗИРОВАТЬ УПРАВЛЯЮЩИЙ ПОРТ 8255

        ;РАЗРЕШИТЬ ПРЕРЫВАНИЯ 8255
        MVI     A,PAIE              ;РАЗРЕШИТЬ ПРЕРЫВАНИЯ ОТ ПОРТА A
```

```
        OUT     PPICTRL
        MVI     A,PBIE              ;РАЗРЕШИТЬ ПРЕРЫВАНИЯ ОТ ПОРТА В
        OUT     PPICTRL

        EI                          ;РАЗРЕШИТЬ ПРЕРЫВАНИЯ
        RET

        ;ДИСПЕТЧЕР ПРЕРЫВАНИЙ ПО ВВОДУ (ЧТЕНИЮ)
RDHDLR:
        PUSH    PSW                 ;СОХРАНИТЬ AF
        IN      PPIA                ;СЧИТАТЬ ДАННЫЕ С 8255
        STA     RECDAT              ;СОХРАНИТЬ ДАННЫЕ В БУФЕРЕ ВВОДА
        MVI     A,0FFH
        STA     RECDF               ;УКАЗАТЬ, ЧТО В БУФЕРЕ ВВОДА ЕСТЬ ДАННЫЕ
        MVI     A,EOI               ;ОЧИСТИТЬ ПРЕРЫВАНИЯ 8259
        OUT     PIC0
        POP     PSW                 ;ВОССТАНОВИТЬ AF
        EI                          ;ВНОВЬ РАЗРЕШИТЬ ПРЕРЫВАНИЯ
        RET

        ;ДИСПЕТЧЕР ПРЕРЫВАНИЙ ПО ВЫВОДУ (ЗАПИСИ)
WRHDLR:
        PUSH    PSW                 ;СОХРАНИТЬ РЕГИСТР А
        LDA     TRNDF               ;ПРОВЕРИТЬ ФЛАГ НАЛИЧИЯ ДАННЫХ
        ORA     A
        JZ      NODATA              ;ПЕРЕЙТИ, ЕСЛИ НЕТ ДАННЫХ ДЛЯ ПЕРЕДАЧИ
        CALL    OUTDAT              ;ПОСЛАТЬ ДАННЫЕ В 8255
        JMP     WRDONE

        ;ЕСЛИ ПРОИЗОШЛО ПРЕРЫВАНИЕ ПРИ ОТСУТСТВИИ ДАННЫХ, МЫ ДОЛЖНЫ
        ; ЕГО ОЧИСТИТЬ (В 8259), ЧТОБЫ ИЗБЕЖАТЬ ЗАЦИКЛИВАНИЯ. ПОЗЖЕ,
        ; КОГДА СИМВОЛ СТАНОВИТСЯ ДОСТУПНЫМ, НЕОБХОДИМО ЗНАТЬ, ЧТО БЫЛО
        ; ПРЕРЫВАНИЕ ПО ВЫВОДУ, КОТОРОЕ ЕЩЕ НЕ ОБСЛУЖИВАЛОСЬ. ЭТО МОЖНО
        ; ОПРЕДЕЛИТЬ ПО ФЛАГУ OIE ОЖИДАНИЯ ПРЕРЫВАНИЯ ПО ВЫВОДУ. КОГДА
        ; ПРЕРЫВАНИЕ ПРОИСХОДИТ, НО ОСТАЕТСЯ НЕОБСЛУЖЕННЫМ, ЭТОТ ФЛАГ
        ; ОЧИЩАЕТСЯ. КРОМЕ ТОГО, ОН ОЧИЩАЕТСЯ С САМОГО НАЧАЛА, ТАК КАК
        ; 8255 НАЧИНАЕТ РАБОТАТЬ С ГОТОВНОСТЬЮ ПО ВЫВОДУ. OIE
        ; УСТАНАВЛИВАЕТСЯ КАЖДЫЙ РАЗ, КОГДА ДАННЫЕ В ДЕЙСТВИТЕЛЬНОСТИ
        ; ПОСЫЛАЮТСЯ НА 8255. ТАКИМ ОБРАЗОМ, ПОДПРОГРАММА OUTCH МОЖЕТ
        ; ПРОВЕРИТЬ OIE, ЧТОБЫ ОПРЕДЕЛИТЬ, СЛЕДУЕТ ЛИ ПОСЫЛАТЬ ДАННЫЕ
        ; НЕМЕДЛЕННО ИЛИ ЖДАТЬ ПРЕРЫВАНИЯ ПО ВЫВОДУ.
        ;ПРОБЛЕМА ЗДЕСЬ СОСТОИТ В ТОМ, ЧТО УСТРОЙСТВО ВЫВОДА МОЖЕТ
        ; ЗАПРОСИТЬ ОБСЛУЖИВАНИЕ ДО ТОГО, КАК У ЭВМ ПОЯВИТСЯ ЧТО-ЛИБО
        ; ДЛЯ ПЕРЕДАЧИ УСТРОЙСТВУ (В ОТЛИЧИЕ ОТ УСТРОЙСТВА ВВОДА,
        ; КОТОРОЕ, ЗАПРАШИВАЯ ОБСЛУЖИВАНИЕ, ИМЕЕТ ДАННЫЕ). ЭТА ПРОБЛЕМА
        ; НЕОБСЛУЖЕННЫХ ПРЕРЫВАНИЙ ПО ВЫВОДУ РЕШАЕТСЯ С ПОМОЩЬЮ
        ; МНОГОКРАТНОЙ УСТАНОВКИ ФЛАГА OIE, ПОЗВОЛЯЮЩЕГО РАСПОЗНАВАТЬ
        ; ЭТИ ПРЕРЫВАНИЯ.
NODATA:
        SUB     A
        STA     OIE                 ;ПРЕРЫВАНИЯ НЕ ОЖИДАЕТСЯ

WRDONE:
        MVI     A,EOI               ;ОЧИСТИТЬ ПРЕРЫВАНИЕ НА 8259
        OUT     PIC0
        POP     PSW                 ;ВОССТАНОВИТЬ AF
        EI                          ;ВНОВЬ РАЗРЕШИТЬ ПРЕРЫВАНИЯ
        RET
```

```
        ;*******************************************
        ;ПОДПРОГРАММА: OUTDAT
        ;НАЗНАЧЕНИЕ:   ПОСЫЛАЕТ СИМВОЛ 8255
        ;ВХОД:  TRNDAT = СИМВОЛ
        ;ВЫХОД: НЕТ ПАРАМЕТРОВ
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF
        ;*******************************************
OUTDAT:
        LDA     TRNDAT          ;ВЗЯТЬ ДАННЫЕ ИЗ БУФЕРА ВЫВОДА
        OUT     PPIB            ;ПОСЛАТЬ ДАННЫЕ 8255
        SUB     A               ;ОТМЕТИТЬ, ЧТО БУФЕР ВЫВОДА ПУСТОЙ
        STA     TRNDF
        DCR     A               ;ОТМЕТИТЬ, ЧТО ОЖИДАЕТСЯ ПРЕРЫВАНИЕ ПО
        STA     OIE             ; ВЫВОДУ, OIE = ШЕСТНАДЦАТЕРИЧНОЕ
                                ; ЧИСЛО FF
        RET

        ;СЕКЦИЯ ДАННЫХ
RECDAT: DS      1               ;ПОЛУЧАЕМЫЕ ДАННЫЕ
RECDF:  DS      1               ;ФЛАГ ПОЛУЧАЕМЫХ ДАННЫХ
                                ; (0 = НЕТ ДАННЫХ, FF = ДАННЫЕ ЕСТЬ)
TRNDAT: DS      1               ;ПЕРЕДАВАЕМЫЕ ДАННЫЕ
TRNDF:  DS      1               ;ФЛАГ ПЕРЕДАВАЕМЫХ ДАННЫХ
                                ; (0 = БУФЕР ПУСТОЙ, FF = БУФЕР ПОЛНЫЙ)
OIE:    DS      1               ;ФЛАГ ОЖИДАНИЯ ПРЕРЫВАНИЯ ПО ВЫВОДУ
                                ; (0 = ПРЕРЫВАНИЯ НЕ ОЖИДАЕТСЯ,
                                ; FF = ПРЕРЫВАНИЕ ОЖИДАЕТСЯ)


;                                                                    ;
;                                                                    ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                            ;
;                                                                    ;
;                                                                    ;

        ;ЭКВИВАЛЕНТНЫЕ СИМВОЛЫ
ESCAPE  EQU     1BH             ;СИМВОЛ ASCII ESCAPE
TESTCH  EQU     'A'             ;СИМВОЛ ДЛЯ ТЕСТА = A

SC11B:
        CALL    INIT            ;ИНИЦИАЛИЗИРОВАТЬ 8255, СИСТЕМУ
                                ; ПРЕРЫВАНИЙ

        ;ПРОСТОЙ ПРИМЕР - ЧИТАТЬ И ВЫДАВАТЬ СИМВОЛЫ, ПОКА НЕ БУДЕТ
        ; ПОЛУЧЕН СИМВОЛ ESC
LOOP:
        CALL    INCH            ;ПРОЧИТАТЬ СИМВОЛ
        PUSH    PSW
        CALL    OUTCH           ;ВЫДАТЬ СИМВОЛ
        POP     PSW
        CPI     ESCAPE          ;СИМВОЛ ESCAPE?
        JNZ     LOOP            ;ЕСЛИ НЕТ, ОСТАТЬСЯ В ЦИКЛЕ

        ;ПРИМЕР АСИНХРОННОЙ РАБОТЫ
        ; ВЫДАВАТЬ НА КОНСОЛЬ "A", НО ОДНОВРЕМЕННО СЛЕДИТЬ И ЗА
        ; СТОРОНОЙ ВХОДА, ЧИТАЯ И ВЫДАВАЯ ЛЮБЫЕ ВВОДИМЫЕ СИМВОЛЫ
```

```
ASYNLP:
        ;ЕСЛИ ВЫВОД НЕ ЗАНЯТ, ВЫВОДИТЬ "A"
        CALL    OUTST           ;ВЫВОД ЗАНЯТ?
        JC      ASYNLP          ;ЕСЛИ ДА, ПЕРЕЙТИ
        MVI     A,TESTCH
        CALL    OUTCH           ;ВЫВЕСТИ ТЕСТОВЫЙ СИМВОЛ

        ;ПРОВЕРИТЬ ПОРТ ВВОДА
        ;ВЫВЕСТИ СИМВОЛ, ЕСЛИ ОН ЕСТЬ
        ;ВЫЙТИ ПО СИМВОЛУ ESCAPE
        CALL    INST            ;ЕСТЬ ДАННЫЕ НА ВХОДЕ?
        JC      ASYNLP          ;ЕСЛИ НЕТ, ПЕРЕЙТИ (ПОСЛАТЬ ЕЩЕ ОДИН
                                ; СИМВОЛ "A")
        CALL    INCH            ;ВЗЯТЬ СИМВОЛ
        CPI     ESCAPE          ;ЭТО ESCAPE?
        JZ      DONE            ;ЕСЛИ ДА, ПЕРЕЙТИ
        CALL    OUTCH           ;ИНАЧЕ ВЫДАТЬ СИМВОЛ
        JMP     ASYNLP          ;И ПРОДОЛЖИТЬ РАБОТУ

DONE:
        JMP     SC11B

        END
```

11С. БУФЕРИРОВАННЫЙ ВВОД-ВЫВОД С ИСПОЛЬЗОВАНИЕМ ПРОГРАММИРУЕМОГО ИНТЕРФЕЙСА СВЯЗИ 8251 (SINTB)

Выполняются ввод и вывод по прерываниям с использованием 8251 и многосимвольных буферов. Содержатся следующие подпрограммы:

1) INCH – читает символ из буфера ввода;
2) INST – определяет, пустой ли буфер ввода;
3) OUTCH – записывает символ в буфер вывода;
4) OUTST – определяет, заполнен ли буфер вывода;
5) INIT – инициализирует буферы, систему прерываний и интерфейс 8251.

Прерывания обслуживаются следующими подпрограммами:

1) RDHDLR – отвечает на прерывание по вводу, считывая символ с 8251 в буфер ввода;
2) WRHDLR – отвечает на прерывание по выводу, записывая символ из буфера вывода в 8251.

Процедура.

1. INCH – проверяется в цикле, не стал ли символ доступным, получается символ из начала буфера ввода, начало буфера перемещается вниз на одну позицию и счетчик буфера ввода уменьшается на 1.

2. INST – флаг переноса очищается, если счетчик буфера ввода равен 0, и устанавливается в противном случае.

3. OUTCH – проверяется в цикле, не освободилось ли место в буфере вывода (т. е. когда буфер не будет полным), запоминается символ в конце буфера, конец пересылается вверх на одну позицию и счетчик буфера вывода увеличивается на 1.

4. OUTST – устанавливается флаг переноса, если счетчик буфера вывода равен длине буфера (т. е., если буфер вывода полный), и очищается флаг переноса в противном случае.

5. INIT – очищаются счетчики буферов, устанавливаются начала и концы буферов равными базовым адресам буферов, устанавливаются векторы прерываний и инициализируется 8251 с записью соответствующих значений в его командный регистр и регистр способа работы. Для более детального ознакомления с инициализацией 8251 см. подпрограмму 10E. Кроме того, INIT очищает флаг ожидания прерывания по выводу, указывающий, что программируемый интерфейс изначально готов к передаче данных.

6. RDHDLR – символ считывается с 8251. Если есть место в буфере ввода, то запоминается символ в конце буфера, начало перемещается на одну позицию вверх и счетчик буфера ввода увеличивается на 1. Если буфер заполнен, то просто отбрасывается символ.

7. WRHDLR – определяется, есть ли готовые данные для вывода. Если данных нет, то просто очищается прерывание по выводу. Если данные доступны, то получается символ из начала буфера вывода, пересылается начало вверх на одну позицию и на 1 уменьшается счетчик буфера вывода.

Новая проблема, связанная с многосимвольными буферами, состоит в управлении очередями. Чтение данных главной программой должно осуществляться в том порядке, в котором их получает подпрограмма обслуживания прерываний по вводу. Аналогично подпрограммой обслуживания прерываний по выводу должны посылаться данные в порядке их записи главной программой. Таким образом, подпрограммами ввода обусловлены следующие требования:

1) главная программа должна знать, свободен ли буфер ввода;

2) если буфер ввода не пустой, главная программа должна содержать информацию о нахождении самого старого символа (т. е. символа, который был получен первым);

3) подпрограмма, обслуживающая прерывания по вводу, должна содержать информацию о степени заполнения буфера ввода;

4) если буфер ввода не полный, подпрограмма, обслуживающая прерывания по вводу, должна содержать информацию о том, где есть следующее свободное место (т. е. где может быть запомнен следующий символ).

К буферу вывода подпрограмма обслуживания прерываний по выводу и главная программа предъявляют аналогичные требования, хотя в этом случае меняются роли посылающей и принимающей сторон.

Требования 1 и 3 удовлетворяются с помощью счетчика ICNT. INIT устанавливает ICNT в 0, каждый раз при получении символа подпрограммой обслуживания прерываний к счетчику добавляется 1 (считается, что буфер не полный), а главной программой каждый раз при удалении символа из буфера счетчик уменьшается на 1. Таким образом, при проверке ICNT на 0 главной программой может определяться, пустой ли буфер. Подобным же образом при проверке равенства счетчика размеру буфера подпрограммой обслуживания прерываний может определяться, выполнен ли буфер ввода.

Требования 2 и 4 удовлетворяются с помощью двух указателей:

1) ITAIL содержит адрес следующей свободной ячейки в буфере ввода;

2) IHEAD содержит адрес самого старого символа в буфере ввода.

С помощью подпрограммы INIT инициализируются IHEAD и ITAIL, при этом в них записывается адрес буфера ввода. Как только подпрограмма обслуживания прерываний получает символ, он помещается в буфер в ITAIL и

ITAIL перемещается на одну позицию вверх (считается, что буфер не полный). Главной программой прочитанный символ удаляется из буфера из адреса IHEAD, и IHEAD перемещается вверх на одну позицию. Таким образом, IHEAD "преследует" ITAIL по всему буферу, при этом подпрограммой обслуживания прерываний символы записываются с одного конца (с конца буфера), а главной программой они удаляются с другого (с начала буфера).

Занятая часть буфера может начинаться и кончаться в любом месте. Если IHEAD или ITAIL достигают физического конца буфера, то указатель просто переустанавливается на базовый адрес и таким образом создается циклический буфер. Это значит, что занятая часть буфера может начинаться близко от конца (скажем, с байта номер 195 200-байтного буфера) и продолжаться через начало (скажем, до байта номер 10). В этом случае IHEAD будет равняться BASE+194, ITAIL будет равняться BASE+9 и буфер будет занимать 15 символов с адресами от BASE+194 до BASE+199 и от BASE до BASE+8.

Используемые регистры:
1. INCH: AF, C, DE, HL.
2. INST: AF.
3. OUTCH: AF, DE, HL.
4. OUTST: AF.
5. INIT: AF, HL.

Время выполнения:
1. INCH: приблизительно 207 тактов (8080) или 210 тактов (8085), если символ готов.
2. INST: 46 тактов (8080) или 47 тактов (8085).
3. OUTCH: приблизительно 225 тактов (8080) или 220 тактов (8085), если буфер вывода не заполнен и ожидается прерывание по выводу. Приблизительно 157 дополнительных тактов (8080) или 162 дополнительных такта (8085), если не ожидается прерывания по выводу.
4. OUTST: 34 такта (8080 или 8085).
5. INIT: 352 такта (8080 или 8085).
6. RDHDLR: приблизительно 265 тактов (8080) или 267 тактов (8085).
7. WRHDLR: приблизительно 320 тактов (8080) или 324 такта (8085), если буфер вывода не пустой, 159 тактов (8080) или 163 такта (8085), если буфер вывода пустой.

Примечание: то, что время выполнения дано здесь приблизительно, является результатом изменяющегося количества времени, требуемого для обновления указателей буфера в связи с его циклической организацией.

Размер программы: 267 байт.

Память, необходимая для данных: 11 байт в любом месте ОЗУ для начала и конца буферов ввода и вывода (по два байта, начинающихся с адресов IHEAD, ITAIL, OHEAD и OTAIL соответственно), числа символов в буферах (2 байта с адресами ICNT и OCNT) и флага ожидания прерывания по выводу (адрес OIE). Сюда не включены действительные буферы ввода и вывода.

УСЛОВИЯ НА ВХОДЕ

1. INCH: нет параметров.
2. INST: нет параметров.
3. OUTCH: передаваемый символ в регистре A.

4. OUTST: нет параметров.
5. INIT: нет параметров.

УСЛОВИЯ НА ВЫХОДЕ

1. INCH: символ в A.
2. INST: флаг переноса = 0, если буфер ввода пустой, 1 – если не пустой.
3. OUTCH: нет параметров.
4. OUTST: флаг переноса = 0, если буфер вывода не полный, 1 – если полный.
5. INIT: нет параметров

```
;                                                                       ;
;                                                                       ;
;                                                                       ;
;                                                                       ;
;       ЗАГОЛОВОК:      БУФЕРИРОВАННЫЙ ВВОД-ВЫВОД С ИСПОЛЬЗОВАНИЕМ      ;
;                       ПРОГРАММИРУЕМОГО ИНТЕРФЕЙСА СВЯЗИ 8251          ;
;       ИМЯ:            SINTB                                           ;
;                                                                       ;
;                                                                       ;


;                                                                       ;
;       НАЗНАЧЕНИЕ:     ЭТА ПРОГРАММА СОДЕРЖИТ 5 ПОДПРОГРАММ, КОТОРЫЕ   ;
;                       ВЫПОЛНЯЮТ ВВОД И ВЫВОД ПО ПРЕРЫВАНИЯМ С         ;
;                       ИСПОЛЬЗОВАНИЕМ 8251                             ;
;                                                                       ;
;                       INCH                                            ;
;                         ПРОЧИТАТЬ СИМВОЛ                              ;
;                       INST                                            ;
;                         ОПРЕДЕЛИТЬ СОСТОЯНИЕ ВВОДА (ПУСТОЙ ЛИ БУФЕР   ;
;                         ВВОДА)                                        ;
;                       OUTCH                                           ;
;                         ЗАПИСАТЬ СИМВОЛ                               ;
;                       OUTST                                           ;
;                         ОПРЕДЕЛИТЬ СОСТОЯНИЕ ВЫВОДА (ЗАПОЛНЕН ЛИ БУФЕР;
;                         ВЫВОДА)                                       ;
;                       INIT                                            ;
;                         ИНИЦИАЛИЗИРОВАТЬ 8251 И СИСТЕМУ ПРЕРЫВАНИЙ    ;
;                                                                       ;
;       ВХОД:           INCH                                            ;
;                         НЕТ ПАРАМЕТРОВ                                ;
;                       INST                                            ;
;                         НЕТ ПАРАМЕТРОВ                                ;
;                       OUTCH                                           ;
;                         РЕГИСТР А = ПЕРЕДАВАЕМЫЙ СИМВОЛ               ;
;                       OUTST                                           ;
;                         НЕТ ПАРАМЕТРОВ                                ;
;                       INIT                                            ;
;                         НЕТ ПАРАМЕТРОВ                                ;
;                                                                       ;
;       ВЫХОД:          INCH                                            ;
;                         РЕГИСТР А = СИМВОЛ                            ;
;                       INST                                            ;
;                         ФЛАГ ПЕРЕНОСА = 0, ЕСЛИ БУФЕР ВВОДА ПУСТОЙ,   ;
;                         1, ЕСЛИ ЕСТЬ СИМВОЛ                           ;
;                       OUTCH                                           ;
```

```
;                       НЕТ ПАРАМЕТРОВ                        ;
;                     OUTST                                   ;
;                       ФЛАГ ПЕРЕНОСА = 0, ЕСЛИ БУФЕР ВЫВОДА ПУСТОЙ,  ;
;                       1, ЕСЛИ ОН ПОЛНЫЙ                     ;
;                     INIT                                    ;
;                       НЕТ ПАРАМЕТРОВ                        ;
;                                                             ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ:                                ;
;                     INCH                                    ;
;                       AF,C,DE,HL                            ;
;                     INST                                    ;
;                       AF                                    ;
;                     OUTCH                                   ;
;                       AF,DE,HL                              ;
;                     OUTST                                   ;
;                       AF                                    ;
;                     INIT                                    ;
;                       AF,HL                                 ;
;                                                             ;
;       ВРЕМЯ:        INCH                                    ;
;                       ПРИБЛИЗИТЕЛЬНО 207 ТАКТОВ, ЕСЛИ ЕСТЬ СИМВОЛ,  ;
;                       ДЛЯ 8080 И 210 ДЛЯ 8085               ;
;                     INST                                    ;
;                       46 ТАКТОВ ДЛЯ 8080, 47 ДЛЯ 8085       ;
;                     OUTCH                                   ;
;                       ПРИБЛИЗИТЕЛЬНО 225 ТАКТОВ (8080) ИЛИ 223 ТАКТА;
;                       (8085), ЕСЛИ БУФЕР ВЫВОДА НЕ ЗАПОЛНЕН И  ;
;                       ОЖИДАЕТСЯ ПРЕРЫВАНИЕ ПО ВЫВОДУ        ;
;                     OUTST                                   ;
;                       34 ТАКТА ДЛЯ 8080 И 8085              ;
;                     INIT                                    ;
;                       352 ТАКТА ДЛЯ 8080 И 8085             ;
;                     RDHDLR                                  ;
;                       265 ТАКТОВ ДЛЯ 8080, 267 ТАКТОВ ДЛЯ 8085  ;
;                     WRHDLR                                  ;
;                       320 ТАКТОВ ДЛЯ 8080 И 324 ДЛЯ 8085,   ;
;                       ЕСЛИ БУФЕР ВЫВОДА ПОЛНЫЙ              ;

;                                                             ;
;       РАЗМЕР:       ПРОГРАММА - 267 БАЙТ                    ;
;                     ДАННЫЕ    -  11 БАЙТ ПЛЮС РАЗМЕР БУФЕРОВ  ;
;                                                             ;
;                                                             ;

        ;ЭКВИВАЛЕНТНЫЕ ЗНАЧЕНИЯ ДЛЯ ПРОГРАММИРУЕМОГО ИНТЕРФЕЙСА  СВЯЗИ
        ; (PCI) 8251.
        ; 8251 ПРОГРАММИРУЕТСЯ ДЛЯ СЛЕДУЮЩЕГО РЕЖИМА РАБОТЫ:
        ;  АСИНХРОННЫЕ ОПЕРАЦИИ,
        ;  УМНОЖЕНИЯ СКОРОСТИ НА 16,
        ;  8-РАЗРЯДНЫЕ СИМВОЛЫ,
        ;  2 СТОПОВЫХ РАЗРЯДА.
        ;ПРОИЗВОЛЬНЫЕ АДРЕСА ПОРТОВ 8251
USARTDR EQU     0B4H            ;РЕГИСТР ДАННЫХ
USARTSR EQU     0B5H            ;РЕГИСТР СОСТОЯНИЯ
USARTCR EQU     0B5H            ;РЕГИСТР УПРАВЛЕНИЯ

        ;КОМАНДА НАСТРОЙКИ 8251 НА АСИНХРОННЫЙ РЕЖИМ РАБОТЫ
```

```
MODE    EQU     11001110B       ;РАЗРЯДЫ 1,0 = 10 (ПРИЗНАК
                                ; УМНОЖЕНИЯ СКОРОСТИ НА 16)
                                ;РАЗРЯДЫ 3,2 = 11 (8-РАЗРЯДНЫЕ СИМВОЛЫ)
                                ;РАЗРЯД 4 = 0 (БЕЗ РАЗРЯДА ЧЕТНОСТИ)
                                ;РАЗРЯД 5 = 0 (НЕ ИСПОЛЬЗУЕТСЯ)
                                ;РАЗРЯДЫ 7,6 = 11 (2 СТОПОВЫХ РАЗРЯДА)

        ;КОМАНДА АСИНХРОННОЙ РАБОТЫ 8251
CMD     EQU     00010111B       ;РАЗРЯД 0 = 1 (РАЗРЕШИТЬ ПЕРЕДАЧУ)
                                ;РАЗРЯД 1 = 1 (ДАННЫЕ НА ТЕРМИНАЛЕ
                                ; ГОТОВЫ
                                ;РАЗРЯД 2 = 1 (РАЗРЕШИТЬ ПРИЕМ)
                                ;РАЗРЯД 3 = 0 (НЕТ СИМВОЛА BREAK
                                ; (ПРЕРВАТЬ))
                                ;РАЗРЯД 4 = 1 (СБРОСИТЬ ОШИБКУ)
                                ;РАЗРЯД 5 = 0 (РАЗРЕШИТЬ ПЕРЕДАЧУ)
                                ;РАЗРЯД 6 = 0 (НЕТ ЗАПРОСА, КОТОРЫЙ
                                ; ТРЕБУЕТСЯ ПОСЛАТЬ)
                                ;РАЗРЯД 7 = 0 (НЕТ ПРОИЗВОЛЬНОГО
                                ; ПОИСКА)
        ;ЭКВИВАЛЕНТНЫЕ ЗНАЧЕНИЯ ДЛЯ ПРОГРАММИРУЕМОГО КОНТРОЛЛЕРА
        ; ПРЕРЫВАНИЙ (PIC) 8259.
        ; 8259 ПРОГРАММИРУЕТСЯ ДЛЯ СЛЕДУЮЩЕГО РЕЖИМА РАБОТЫ:
        ;  ОДНО УСТРОЙСТВО (В ОТЛИЧИЕ ОТ НЕСКОЛЬКИХ КОНТРОЛЛЕРОВ 8259),
        ;  РЕЖИМ ПОЛНОСТЬЮ ВЛОЖЕННЫХ ПРЕРЫВАНИЯ,
        ;  ВСЕ ПРЕРЫВАНИЯ РАЗРЕШЕНЫ,
        ;  RESTART 4, АДРЕС 0020, ПРЕРЫВАНИЯ ПО ЧТЕНИЮ,
        ;  RESTART 5, АДРЕС 0028, ПРЕРЫВАНИЯ ПО ЗАПИСИ.
        ;ПРОИЗВОЛЬНЫЕ АДРЕСА ПОРТОВ 8259
PIC0    EQU     0A0H            ;ПОРТ 1 PIC
PIC1    EQU     0A1H            ;ПОРТ 2 PIC

        ;ВЕКТОРЫ ПРЕРЫВАНИЙ
RDITRP  EQU     0020H           ;ВЕКТОР ПРЕРЫВАНИЯ ДЛЯ ЧТЕНИЯ
WRITRP  EQU     0028H           ;ВЕКТОР ПРЕРЫВАНИЯ ДЛЯ ЗАПИСИ

        ;КОМАНДНЫЕ СЛОВА ICW1 И ICW2 ДЛЯ ИНИЦИАЛИЗАЦИИ 8259
ICW1    EQU     00010010B       ;РАЗРЯД 0 = 0 (НЕ ИСПОЛЬЗУЕТСЯ)
                                ;РАЗРЯД 1 = 1 (ОДИН КОНТРОЛЛЕР 8259)
                                ;РАЗРЯД 2 = 0 (ИНТЕРВАЛ ВЕКТОРА = 8 БАЙТ)
                                ;РАЗРЯД 3 = 0 (НЕ ИСПОЛЬЗУЕТСЯ)
                                ;РАЗРЯД 4 = 1 (ДОЛЖЕН БЫТЬ РАВЕН 1)
                                ;РАЗРЯДЫ 7,6,5 = 0 (РАЗРЯДЫ 5-7 БАЗОВОГО
                                ; АДРЕСА ДЛЯ КОМАНД RST)
ICW2    EQU     0               ;СТАРШИЙ БАЙТ БАЗОВОГО АДРЕСА ДЛЯ КОМАНД
                                ; RST)

        ;КОМАНДНОЕ СЛОВО ДЛЯ РАБОТЫ 8259
EOI     EQU     00100000B       ;КОНЕЦ КОМАНДНОГО СЛОВА ПРЕРЫВАНИЯ

        ;ПРОЧИТАТЬ СИМВОЛ
INCH:
        CALL    INST            ;ВЗЯТЬ СОСТОЯНИЕ ВВОДА
        JNC     INCH            ;ЖДАТЬ, ЕСЛИ НЕТ СИМВОЛА
        DI                      ;ЗАПРЕТИТЬ ПРЕРЫВАНИЯ
        LXI     H,ICNT          ;УМЕНЬШИТЬ НА 1 СЧЕТЧИК БУФЕРА ВВОДА
        DCR     M
```

```
        LHLD    IHEAD           ;ВЗЯТЬ СИМВОЛ ИЗ НАЧАЛА БУФЕРА ВВОДА
        MOV     C,M
        CALL    INCIPTR         ;УВЕЛИЧИТЬ УКАЗАТЕЛЬ НАЧАЛА БУФЕРА НА 1
        SHLD    IHEAD
        MOV     A,C             ;РЕГИСТР A = ПРОЧИТАННЫЙ СИМВОЛ
        EI                      ;ВНОВЬ РАЗРЕШИТЬ ПРЕРЫВАНИЯ
        RET

        ;ВОЗВРАТИТЬ СОСТОЯНИЕ ВВОДА (ЕСЛИ ЕСТЬ ДАННЫЕ, ФЛАГ ПЕРЕНОСА = 1)
INST:
        LDA     ICNT            ;ПРОВЕРИТЬ СЧЕТЧИК БУФЕРА ВВОДА
        ORA     A
        RZ                      ;ЕСЛИ БУФЕР ПУСТОЙ, ВОЗВРАТИТЬСЯ
                                ; ФЛАГ ПЕРЕНОСА = 0
        STC                     ;УСТАНОВИТЬ ФЛАГ ПЕРЕНОСА, ЧТОБЫ УКАЗАТЬ,
                                ; ЧТО ЕСТЬ ДАННЫЕ
        RET                     ; ВОЗВРАТИТЬСЯ, ФЛАГ ПЕРЕНОСА = 1,
                                ; ДАННЫЕ ЕСТЬ

        ;ЗАПИСАТЬ СИМВОЛ
OUTCH:
        PUSH    PSW             ;СОХРАНИТЬ ЗАПИСЫВАЕМЫЙ СИМВОЛ

        ;ЖДАТЬ, ПОКА БУФЕР ВЫВОДА НЕ БУДЕТ ПУСТОЙ, ЗАПОМНИТЬ
        ; СЛЕДУЮЩИЙ СИМВОЛ
WAITOC:
        CALL    OUTST           ;ВЗЯТЬ СОСТОЯНИЕ ВЫВОДА
        JC      WAITOC          ;ЖДАТЬ, ЕСЛИ БУФЕР ВЫВОДА ПОЛНЫЙ
        DI                      ;ЗАПРЕТИТЬ ПРЕРЫВАНИЯ НА ВРЕМЯ
                                ; ПРОВЕРКИ БУФЕРА И СОСТОЯНИЯ ПРЕРЫВАНИЯ
        LXI     H,OCNT          ;УВЕЛИЧИТЬ НА 1 СЧЕТЧИК БУФЕРА ВЫВОДА
        INR     M
        LHLD    OTAIL           ;УСТАНОВИТЬ УКАЗАТЕЛЬ НА СЛЕДУЮЩУЮ
                                ; СВОБОДНУЮ ЯЧЕЙКУ В БУФЕРЕ
        POP     PSW             ;ВЗЯТЬ СИМВОЛ
        MOV     M,A             ;ЗАПОМНИТЬ СИМВОЛ В КОНЦЕ БУФЕРА
        CALL    INCOPTR         ;УВЕЛИЧИТЬ УКАЗАТЕЛЬ НА 1
        SHLD    OTAIL
        LDA     OIE             ;ПРОВЕРИТЬ ФЛАГ ОЖИДАНИЯ ПРЕРЫВАНИЯ
        ORA     A               ; ПО ВЫВОДУ
        CZ      OUTDAT          ;ЕСЛИ ПЕРЫВАНИЕ ПО ВЫВОДУ НЕ ОЖИДАЕТСЯ,
                                ; НЕМЕДЛЕННО ПОСЛАТЬ СИМВОЛ
        EI                      ;ВНОВЬ РАЗРЕШИТЬ ПРЕРЫВАНИЯ
        RET

        ;СОСТОЯНИЕ ВЫВОДА (ФЛАГ ПЕРЕНОСА = 1, ЕСЛИ БУФЕР ВЫВОДА ПОЛНЫЙ)
OUTST:
        LDA     OCNT            ;ВЗЯТЬ СЧЕТЧИК БУФЕРА ВЫВОДА
        CPI     SZOBUF          ;БУФЕР ВЫВОДА ЗАПОЛНЕН?
        CMC                     ;ИНВЕРТИРОВАТЬ ФЛАГ ПЕРЕНОСА
        RET                     ;ФЛАГ ПЕРЕНОСА = 1, ЕСЛИ БУФЕР ПОЛНЫЙ,
                                ; 0 - ЕСЛИ НЕТ

        ;ИНИЦИАЛИЗИРОВАТЬ 8521 И СИСТЕМУ ПРЕРЫВАНИЙ
INIT:
        DI                      ;ДЛЯ ИНИЦИАЛИЗАЦИИ ЗАПРЕТИТЬ ПРЕРЫВАНИЯ
```

```
        ;ИНИЦИАЛИЗИРОВАТЬ СЧЕТЧИКИ И УКАЗАТЕЛИ БУФЕРОВ, ФЛАГИ ПРЕРЫВАНИЯ

        SUB     A
        STA     ICNT                ;БУФЕР ВВОДА ПУСТОЙ
        STA     OCNT                ;БУФЕР ВЫВОДА ПУСТОЙ
        STA     OIE                 ;УКАЗАТЬ, ЧТО ПРЕРЫВАНИЯ ПО ВЫВОДУ
                                    ; НЕ ОЖИДАЕТСЯ
        LXI     H,IBUF              ;УСТАНОВИТЬ УКАЗАТЕЛЬ НАЧАЛА/КОНЦА
        SHLD    IHEAD               ; ДЛЯ ВВОДА НА ПЕРВЫЙ СИМВОЛ БУФЕРА
        SHLD    ITAIL               ; ВВОДА
        LXI     H,OBUF              ;УСТАНОВИТЬ УКАЗАТЕЛЬ НАЧАЛА/КОНЦА
        SHLD    OHEAD               ; ДЛЯ ВЫВОДА НА ПЕРВЫЙ СИМВОЛ
        SHLD    OTAIL               ; БУФЕРА ВЫВОДА

        ;ИНИЦИАЛИЗИРОВАТЬ ВЕКТОРЫ ПРЕРЫВАНИЙ
        LXI     H,RDHDLR
        MVI     A,0C3H              ;РЕГИСТР A = КОД ОПЕРАЦИИ КОМАНДЫ JMP
        STA     RDITRP              ;ЗАПОМНИТЬ КОД ОПЕРАЦИИ КОМАНДЫ JMP
                                    ; ДЛЯ ПРЕРЫВАНИЙ ПО ЧТЕНИЮ
        SHLD    RDITRP+1            ;ЗАПОМНИТЬ АДРЕС ПЕРЕХОДА
        LXI     H,WRHDLR
        STA     WRITRP              ;ЗАПОМНИТЬ КОД ОПЕРАЦИИ КОМАНДЫ JMP
                                    ; ДЛЯ ПРЕРЫВАНИЙ ПО ЗАПИСИ
        SHLD    WRITRP+1            ;ЗАПОМНИТЬ АДРЕС ПЕРЕХОДА

        ;ИНИЦИАЛИЗИРОВАТЬ КОНТРОЛЛЕР ПРЕРЫВАНИЙ 8259
        MVI     A,ICW1              ;ПОСЛАТЬ ПЕРВОЕ СЛОВО В ПОРТ 0 PIC
        OUT     PIC0
        MVI     A,ICW2              ;ПОСЛАТЬ ВТОРОЕ СЛОВО В ПОРТ 1 PIC
        OUT     PIC1

        ;ИНИЦИАЛИЗИРОВАТЬ 8251
        MVI     A,80H               ;СБРОСИТЬ ПРОГРАММНО 8251,
        OUT     USARTCR             ; ПОСЛАВ ЕМУ 2 БАЙТА, СОДЕРЖАЩИХ
        OUT     USARTCR             ; ШЕСТНАДЦАТЕРИЧНОЕ ЗНАЧЕНИЕ 80,
        MVI     A,40H               ; ЗА КОТОРЫМИ СЛЕДУЕТ КОМАНДА СБРОСА
        OUT     USARTCR
        MVI     A,MODE              ;ВЫДАТЬ БАЙТ РЕЖИМА РАБОТЫ
        OUT     USARTCR
        MVI     A,CMD               ;ВЫДАТЬ КОМАНДНЫЙ БАЙТ
        OUT     USARTCR
        IN      USARTDR             ;СЧИТАТЬ РЕГИСТР ДАННЫХ, ЧТОБЫ
                                    ; ИЗНАЧАЛЬНО ОЧИСТИТЬ РЕГИСТР
                                    ; ВВОДА ОТ СЛУЧАЙНЫХ ДАННЫХ

        EI                          ;РАЗРЕШИТЬ ПРЕРЫВАНИЯ
        RET

        ;ДИСПЕТЧЕР ПРЕРЫВАНИЙ ПО ВВОДУ (ЧТЕНИЮ)
RDHDLR:
        PUSH    PSW                 ;СОХРАНИТЬ РЕГИСТРЫ
        PUSH    B
        PUSH    D
        PUSH    H
        IN      USARTDR             ;ПРОЧИТАТЬ ДАННЫЕ С 8251
        MOV     C,A                 ;СОХРАНИТЬ ДАННЫЕ В РЕГИСТРЕ C
        LXI     H,ICNT              ;ЕСТЬ МЕСТО В БУФЕРЕ ВВОДА?
        MOV     A,M
```

```
        CPI     SZIBUF
        JNC     XITRH           ;ПЕРЕЙТИ, ЕСЛИ НЕТ МЕСТА В БУФЕРЕ ВВОДА
        INR     M               ;УВЕЛИЧИТЬ СЧЕТЧИК БУФЕРА ВВОДА
        LHLD    ITAIL           ;ЗАПОМНИТЬ СИМВОЛ В КОНЦЕ БУФЕРА ВВОДА
        MOV     M,C
        CALL    INCIPTR         ;УВЕЛИЧИТЬ УКАЗАТЕЛЬ КОНЦА
        SHLD    ITAIL

XITRH:  POP     H               ;ВОССТАНОВИТЬ РЕГИСТРЫ
        POP     D
        POP     B
        MVI     A,EOI           ;ОЧИСТИТЬ ПРЕРЫВАНИЯ 8259
        OUT     PICO
        POP     PSW
        EI                      ;ВНОВЬ РАЗРЕШИТЬ ПРЕРЫВАНИЯ
        RET

        ;ДИСПЕТЧЕР ПРЕРЫВАНИЙ ПО ВЫВОДУ (ЗАПИСИ)
WRHDLR:
        PUSH    PSW             ;СОХРАНИТЬ РЕГИСТРЫ
        PUSH    B
        PUSH    D
        PUSH    H

        LDA     OCNT            ;ПРОВЕРИТЬ СЧЕТЧИК БУФЕРА ВЫВОДА
        ORA     A
        JZ      NODATA          ;ПЕРЕЙТИ, ЕСЛИ НЕТ ДАННЫХ ДЛЯ ПЕРЕДАЧИ
        CALL    OUTDAT          ;ИНАЧЕ ПОСЛАТЬ ДАННЫЕ В 8251
        JMP     WRDONE

        ;ЕСЛИ ПРОИЗОШЛО ПРЕРЫВАНИЕ ПРИ ОТСУТСТВИИ ДАННЫХ, НЕОБХОДИМО
        ; ЕГО ОЧИСТИТЬ, ЧТОБЫ ИЗБЕЖАТЬ ЗАЦИКЛИВАНИЯ. КОГДА СЛЕДУЮЩИЙ
        ; СИМВОЛ ГОТОВ, ОН ДОЛЖЕН БЫТЬ НЕМЕДЛЕННО ПОСЛАН, ТАК КАК
        ; ПРЕРЫВАНИЯ НЕ БУДЕТ. ЭТО СОСТОЯНИЕ, ПРИ КОТОРОМ ПРЕРЫВАНИЕ
        ; ПО ВЫВОДУ УЖЕ ПРОИЗОШЛО, НО НЕ БЫЛО ОБСЛУЖЕНО, ОТМЕЧАЕТСЯ
        ; ОЧИСТКОЙ ФЛАГА OIE (ФЛАГ ОЖИДАНИЯ ПРЕРЫВАНИЯ ПО ВЫВОДУ).
NODATA:
        SUB     A
        STA     OIE             ;ПРЕРЫВАНИЯ НЕ ОЖИДАЕТСЯ

WRDONE:
        POP     H               ;ВОССТАНОВИТЬ РЕГИСТРЫ
        POP     D
        POP     B
        MVI     A,EOI           ;ОЧИСТИТЬ ПРЕРЫВАНИЕ НА 8259
        OUT     PICO
        POP     PSW
        EI                      ;ВНОВЬ РАЗРЕШИТЬ ПРЕРЫВАНИЯ
        RET

        ;*******************************************
        ;ПОДПРОГРАММА: OUTDAT
        ;НАЗНАЧЕНИЕ:   ПОСЛАТЬ СИМВОЛ 8251
        ;ВХОД:  TRNDAT = СИМВОЛ
        ;ВЫХОД: НЕТ ПАРАМЕТРОВ
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF,DE,HL
        ;*******************************************
```

```
OUTDAT:
        LHLD    OHEAD
        MOV     A,M             ;ВЗЯТЬ ДАННЫЕ ИЗ НАЧАЛА БУФЕРА ВЫВОДА
        OUT     USARTDR         ;ПОСЛАТЬ ДАННЫЕ 8251
        CALL    INCOPTR         ;УВЕЛИЧИТЬ УКАЗАТЕЛЬ НАЧАЛА
        SHLD    OHEAD
        LXI     H,OCNT          ;УМЕНЬШИТЬ СЧЕТЧИК БУФЕРА ВЫВОДА
        DCR     M
        MVI     A,OFFH          ;ОЖИДАТЬ ПРЕРЫВАНИЯ ПО ВЫВОДУ
        STA     OIE
        RET

        ;******************************************
        ;ПОДПРОГРАММА: INCIPTR
        ;НАЗНАЧЕНИЕ:   УВЕЛИЧИТЬ ПО КОЛЬЦУ УКАЗАТЕЛЬ
        ;              В БУФЕРЕ ВВОДА
        ;ВХОД: HL = УКАЗАТЕЛЬ
        ;ВЫХОД: HL = УКАЗАТЕЛЬ, УВЕЛИЧЕННЫЙ ПО КОЛЬЦУ
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF,DE,HL
        ;******************************************
INCIPTR:
        INX     H               ;УВЕЛИЧИТЬ УКАЗАТЕЛЬ
        LXI     D,EIBUF         ;СРАВНИТЬ УКАЗАТЕЛЬ С КОНЦОМ БУФЕРА
        MOV     A,L
        CMP     E
        RNZ                     ;ВОЗВРАТИТЬСЯ, ЕСЛИ МЛАДШИЕ БАЙТЫ
                                ; НЕ РАВНЫ
        MOV     A,H
        CMP     D
        RNZ                     ;ВОЗВРАТИТЬСЯ, ЕСЛИ СТАРШИЕ БАЙТЫ
                                ; НЕ РАВНЫ
        LXI     H,IBUF          ;ЕСЛИ УКАЗАТЕЛЬ ПОПАДАЕТ НА КОНЕЦ
                                ; БУФЕРА, ТО УСТАНОВИТЬ ЕГО НА
        RET                     ; БАЗОВЫЙ АДРЕС

        ;******************************************
        ;ПОДПРОГРАММА: INCOPTR
        ;НАЗНАЧЕНИЕ:   УВЕЛИЧИТЬ ПО КОЛЬЦУ УКАЗАТЕЛЬ
        ;              В БУФЕРЕ ВЫВОДА
        ;ВХОД: HL = УКАЗАТЕЛЬ
        ;ВЫХОД: HL = УКАЗАТЕЛЬ, УВЕЛИЧЕННЫЙ ПО КОЛЬЦУ
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF,DE,HL
        ;******************************************
INCOPTR:
        INX     H               ;УВЕЛИЧИТЬ УКАЗАТЕЛЬ
        LXI     D,EOBUF         ;СРАВНИТЬ УКАЗАТЕЛЬ С КОНЦОМ БУФЕРА
        MOV     A,L
        CMP     E
        RNZ                     ;ВОЗВРАТИТЬСЯ, ЕСЛИ МЛАДШИЕ БАЙТЫ
                                ; НЕ РАВНЫ
        MOV     A,H
        CMP     D
        RNZ                     ;ВОЗВРАТИТЬСЯ, ЕСЛИ СТАРШИЕ БАЙТЫ
                                ; НЕ РАВНЫ
        LXI     H,OBUF          ;ЕСЛИ УКАЗАТЕЛЬ ПОПАДАЕТ НА КОНЕЦ
                                ; БУФЕРА, ТО УСТАНОВИТЬ ЕГО НА
        RET                     ; БАЗОВЫЙ АДРЕС
```

```
        ;СЕКЦИЯ ДАННЫХ
IHEAD:  DS      2                       ;УКАЗАТЕЛЬ НА САМЫЙ СТАРЫЙ СИМВОЛ В
                                        ; В БУФЕРЕ ВВОДА (СЛЕДУЮЩИЙ СИМВОЛ
                                        ; ДЛЯ ЧТЕНИЯ)
ITAIL:  DS      2                       ;УКАЗАТЕЛЬ НА САМЫЙ НОВЫЙ СИМВОЛ
                                        ; В БУФЕРЕ ВВОДА (ПОСЛЕДНИЙ ПРОЧИТАННЫЙ
                                        ; СИМВОЛ)
ICNT:   DS      1                       ;ЧИСЛО СИМВОЛОВ В БУФЕРЕ ВВОДА
OHEAD:  DS      2                       ;УКАЗАТЕЛЬ НА САМЫЙ СТАРЫЙ СИМВОЛ В
                                        ; В БУФЕРЕ ВЫВОДА (ПОСЛЕДНИЙ ЗАПИСАННЫЙ
                                        ; СИМВОЛ)
OTAIL:  DS      2                       ;УКАЗАТЕЛЬ НА САМЫЙ НОВЫЙ СИМВОЛ
                                        ; В БУФЕРЕ ВЫВОДА (СЛЕДУЮЩИЙ ПОСЫЛАЕМЫЙ
                                        ; СИМВОЛ)
OCNT:   DS      1                       ;ЧИСЛО СИМВОЛОВ В БУФЕРЕ ВЫВОДА
SZIBUF  EQU     10                      ;РАЗМЕР БУФЕРА ВВОДА
IBUF:   DS      SZIBUF                  ;БУФЕР ВВОДА
EIBUF   EQU     *                       ;КОНЕЦ БУФЕРА ВВОДА
SZOBUF  EQU     10                      ;РАЗМЕР БУФЕРА ВЫВОДА
OBUF:   DS      SZOBUF                  ;БУФЕР ВЫВОДА
EOBUF   EQU     *                       ;КОНЕЦ БУФЕРА ВЫВОДА
OIE:    DS      1                       ;ФЛАГ ОЖИДАНИЯ ПРЕРЫВАНИЯ ПО ВЫВОДУ
                                        ; (0 = ПРЕРЫВАНИЯ НЕ ОЖИДАЕТСЯ,
                                        ; FF = ПРЕРЫВАНИЕ ОЖИДАЕТСЯ)


;                                                                       ;
;                                                                       ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                               ;
;                                                                       ;
;                                                                       ;

        ;ЭКВИВАЛЕНТНЫЕ СИМВОЛЫ
ESCAPE  EQU     1BH                     ;СИМВОЛ ASCII ESCAPE
TESTCH  EQU     'A'                     ;СИМВОЛ ДЛЯ ТЕСТА = A

SC11C:
        CALL    INIT                    ;ИНИЦИАЛИЗИРОВАТЬ 8251, СИСТЕМУ
                                        ; ПРЕРЫВАНИЙ

        ;ПРОСТОЙ ПРИМЕР - ЧИТАТЬ И ВЫДАВАТЬ СИМВОЛЫ, ПОКА НЕ БУДЕТ
        ; ПОЛУЧЕН СИМВОЛ ESC
LOOP:
        CALL    INCH                    ;ПРОЧИТАТЬ СИМВОЛ
        PUSH    PSW
        CALL    OUTCH                   ;ВЫДАТЬ СИМВОЛ
        POP     PSW
        CPI     ESCAPE                  ;СИМВОЛ ESCAPE?
        JNZ     LOOP                    ;ЕСЛИ НЕТ, ОСТАТЬСЯ В ЦИКЛЕ

        ;ПРИМЕР АСИНХРОННОЙ РАБОТЫ
        ; ВЫДАВАТЬ НА КОНСОЛЬ "A", НО ОДНОВРЕМЕННО СЛЕДИТЬ И ЗА
        ; СТОРОНОЙ ВХОДА, ЧИТАЯ И ВЫДАВАЯ ЛЮБЫЕ ВВОДИМЫЕ СИМВОЛЫ
ASYNLP:
        ;ЕСЛИ ВЫВОД НЕ ЗАНЯТ, ВЫВОДИТЬ "A"
        CALL    OUTST                   ;ВЫВОД ЗАНЯТ?
        JC      ASYNLP                  ;ЕСЛИ ДА, ПЕРЕЙТИ
```

```
        MVI     A,TESTCH
        CALL    OUTCH           ;ВЫВЕСТИ ТЕСТОВЫЙ СИМВОЛ

        ;ПРОВЕРИТЬ ПОРТ ВВОДА
        ;ВЫВЕСТИ СИМВОЛ, ЕСЛИ ОН ЕСТЬ
        ;ВЫЙТИ ПО СИМВОЛУ ESCAPE
        CALL    INST            ;ЕСТЬ ДАННЫЕ НА ВХОДЕ?
        JC      ASYNLP          ;ЕСЛИ НЕТ, ПЕРЕЙТИ (ПОСЛАТЬ ЕЩЕ ОДИН
                                ; СИМВОЛ "A")
        CALL    INCH            ;ВЗЯТЬ СИМВОЛ
        CPI     ESCAPE          ;ЭТО ESCAPE?
        JZ      DONE            ;ЕСЛИ ДА, ПЕРЕЙТИ
        CALL    OUTCH           ;ИНАЧЕ ВЫДАТЬ СИМВОЛ
        JMP     ASYNLP          ;И ПРОДОЛЖИТЬ РАБОТУ

DONE:
        JMP     SC11C

        END
```

11D. ЧАСЫ И КАЛЕНДАРЬ РЕАЛЬНОГО ВРЕМЕНИ (CLOCK)

Поддерживаются круглосуточные часы, указывающие время суток и рассчитанные на 24 часа, и календарь на основе временных прерываний реального времени, генерируемых программируемым интервальным таймером (PIT) 8253. Содержит следующие подпрограммы:

1) CLOCK – возвращает базовый адрес переменных времени;

2) ICLK – инициализирует прерывания по времени и переменные времени;

3) CLKINT – обновляет время после каждого прерывания (считается,что прерывания должны происходить отдельно на каждый импульс сигнала времени).

Процедура.

1. CLOCK – загружается базовый адрес переменных времени в регистры H и L. Переменные времени запоминаются в следующем порядке (младшие адреса идут первыми): импульсы сигналов времени, секунды, минуты, часы, день, месяц, младший по значению байт года, старший по значению байт года.

2. ICLK – инициализируется 8253, система прерываний и переменные времени. Произвольно выбранное начальное время 00: 00.00, январь 1, 1980. В случае практического применения, разумеется, требуется загрузить или изменить часы извне.

3. CLKINT – уменьшается на 1 оставшийся счетчик интервалов времени и, если необходимо, обновляется остальная часть переменных времени. Конечно, число секунд и минут должно быть меньше 60, а число часов меньше 24. Порядковый номер дня месяца должен быть меньше порядкового номера последнего дня текущего месяца или равен ему. Массив последних дней для каждого месяца начинается по адресу LASTDY.

Если месяц – февраль (т. е. месяц номер 2), то проверяется, является ли данный год високосным. При этом проверяется, равны ли 0 оба младших разряда ячейки памяти YEAR. Если текущий год високосный, то последний день февраля будет 29, а не 28.

Номер месяца не может превышать 12 (декабрь), иначе необходим перенос в номер года. Когда происходят подобные переносы, должны правильно

переинициализироваться переменные, т. е. TICK в DTICK, секунды, минуты и часы в 0, день и месяц в 1 (означающие, соответственно, первый день и январь).

Используемые регистры:
1. CLOCK: HL.
2. ICLK: AF, HL.
3. CLKINT: отсутствуют.
Время выполнения:
1. CLOCK: 20 тактов (8080 или 8085).
2. ICLK: 269 тактов (8080 или 8085).
3. CLKINT: 103 такта (8080) или 105 тактов (8085), если необходимо только уменьшить TICK; максимальное 522 такта (8080) или 510 тактов (8085) при переходе на новый год.
Размер программы: 187 байт.
Память, необходимая для даниых: 8 байт в любом месте ОЗУ для переменных времени (начиная с адреса CLKVAR).

УСЛОВИЯ НА ВХОДЕ

1. CLOCK: нет параметров.
2. ICLK: нет параметров.
3. CLKINT: нет параметров.

УСЛОВИЯ НА ВЫХОДЕ

1. CLOCK: базовый адрес переменных времени в регистрах H и L.
2. ICLK: нет параметров.
3. CLKINT: нет параметров.

ПРИМЕРЫ

В этих примерах считается, что скорость импульсов сигналов времени равна DTICK Гц (меньше 256 Гц; обычно это 60 Гц или 100 Гц) и что данные о времени и данные календаря хранятся в ячейках памяти:

TICK — число интервалов времени до переноса, считаемое вниз от DTICK,
SEC — секунды (от 0 до 59),
MIN — минуты (от 0 до 59),
HOUR — час дня (от 0 до 23),
DAY — день месяца (от 1 до 28, 29, 30 или 31),
MONTH — месяц года (от 1 до 12),
YEAR и YEAR+1 — текущий год.

1. Даниые: март 7, 1982, 11 час 59 мин 59 сек пополудни и проходит 1 интервал времени (TICK) = 1;
(SEC) = 59, (HOUR) = 23, (MONTH) = 03,
(MIN) = 59, (DAY) = 07, (YEAR) = 1982.

Результат: март, 8, 1982, 12 час 00 мин 00 сек до полудня и DTICK интервалов времени (TICK) = DTICK;
(SEC) = 0, (HOUR) = 0, (MONTH) = 03,
(MIN) = 0, (DAY) = 08, (YEAR) = 1982.

2. *Данные:* декабрь 31, 1982, 11 час 59 мин 59 сек пополудни и прошел 1 интервал времени (TICK) = 1;
(SEC) = 59, (HOUR) = 23, (MONTH) = 12,
(MIN) = 59, (DAY) = 31, (YEAR) = 1982.

Результат: январь 1, 1983, 12 час 00 мин 00 сек до полудня и DTICK интервалов времени (TICK) = (DTICK);
(SEC) = 0, (HOUR) = 0, (MONTH) = 1,
(MIN) = 0, (DAY) = 1, (YEAR) = 1983.

```
;                                                                     ;
;                                                                     ;
;                                                                     ;
;                                                                     ;
;       ЗАГОЛОВОК:      ЧАСЫ И КАЛЕНДАРЬ РЕАЛЬНОГО ВРЕМЕНИ            ;
;       ИМЯ:            CLOCK                                         ;
;                                                                     ;
;                                                                     ;


;                                                                     ;
;       НАЗНАЧЕНИЕ:     ЭТА ПРОГРАММА ПОДДЕРЖИВАЕТ ЧАСЫ, УКАЗЫВАЮЩИЕ  ;
;                       ВРЕМЯ СУТОК И РАССЧИТАННЫЕ НА 24 ЧАСА, И      ;
;                       КАЛЕНДАРЬ; РАБОТА ПРОГРАММЫ БАЗИРУЕТСЯ НА     ;
;                       ПРЕРЫВАНИЯХ ОТ ПРОГРАММИРУЕМОГО ТАЙМЕРА 8253. ;
;                                                                     ;
;                       CLOCK                                         ;
;                         ВОЗВРАЩАЕТ БАЗОВЫЙ АДРЕС ПЕРЕМЕННЫХ CLOCK   ;
;                       ICLK                                          ;
;                         ИНИЦИАЛИЗИРУЕТ ТАЙМЕР 8253 И ПРЕРЫВАНИЯ ПО  ;
;                         ВРЕМЕНИ                                     ;
;                                                                     ;
;       ВХОД:           CLOCK                                         ;
;                         НЕТ ПАРАМЕТРОВ                              ;
;                       ICLK                                          ;
;                         НЕТ ПАРАМЕТРОВ                              ;
;                                                                     ;
;       ВЫХОД:          CLOCK                                         ;
;                         РЕГИСТР HL = БАЗОВЫЙ АДРЕС ПЕРЕМЕННЫХ CLOCK ;
;                       ICLK                                          ;
;                         НЕТ ПАРАМЕТРОВ                              ;
;                                                                     ;
;       ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: ВСЕ                                    ;
;                                                                     ;
;       ВРЕМЯ:          CLOCK                                         ;
;                         20 ТАКТОВ                                   ;
;                       ICLK                                          ;
;                         269 ТАКТОВ                                  ;
;                       CLKINT                                        ;
;                         ЕСЛИ ОПРЕДЕЛЯЕТСЯ ТОЛЬКО ИСТЕЧЕНИЕ ИНТЕРВАЛА ;
;                         ВРЕМЕНИ, ТО 103 ТАКТА ДЛЯ 8080 И 105 ТАКТОВ ;
;                         ДЛЯ 8085.                                   ;
;                         МАКСИМАЛЬНОЕ ВРЕМЯ ВЫПОЛНЕНИЯ БУДЕТ ПРИ     ;
;                         ПЕРЕХОДЕ НА НОВЫЙ ГОД: 522 ТАКТА ДЛЯ 8080,  ;
;                         510 ТАКТОВ ДЛЯ 8085                         ;
;                                                                     ;
;       РАЗМЕР:         ПРОГРАММА - 187 БАЙТ                          ;
;                       ДАННЫЕ    -   8 БАЙТ                          ;
;                                                                     ;
;                                                                     ;
```

```
        ;  ПРОГРАММИРУЕМЫЙ ИНТЕРВАЛЬНЫЙ ТАЙМЕР (PIT) 8253
        ;
        ;ИНИЦИАЛИЗИРОВАТЬ СЧЕТЧИК 0 8253 В КАЧЕСТВЕ ГЕНЕРАТОРА
        ; ПОСЛЕДОВАТЕЛЬНОСТИ ПРЯМОУГОЛЬНЫХ ИМПУЛЬСОВ С ЧАСТОТОЙ
        ; 100 ГЦ ДЛЯ ИСПОЛЬЗОВАНИЯ В ЧАСАХ, ОТСЧИТЫВАЮЩИХ ВРЕМЯ СУТОК.
        ;ПОСЛЕДОВАТЕЛЬНОСТЬ ПРЯМОУГОЛЬНЫХ ИМПУЛЬСОВ ГЕНЕРИРУЕТСЯ НА
        ; ВЫХОДЕ 10 8253, КОТОРЫЙ СВЯЗАН С ВЫХОДОМ 21 ПРОГРАММИРУЕМОГО
        ; КОНТРОЛЛЕРА ПРЕРЫВАНИЙ (PIC) 8259.
        ;ТАКИМ ОБРАЗОМ, ПРЕРЫВАНИЯ ПО ВРЕМЕНИ ПРИВЯЗАНЫ К ВЕКТОРУ 3
        ; ПРЕРЫВАНИЙ.
        ;МЫ ПРИНИМАЕМ, ЧТО НА ВЫХОД 18 8253 ПОДАЮТСЯ ТАКТОВЫЕ ИМПУЛЬСЫ
        ; С ЧАСТОТОЙ 4 МГЦ, ТАК ЧТО ЗНАЧЕНИЕ СЧЕТЧИКА, НЕОБХОДИМОЕ ДЛЯ
        ; ГЕНЕРАЦИИ ПОСЛЕДОВАТЕЛЬНОСТИ ПРЯМОУГОЛЬНЫХ ИМПУЛЬСОВ С
        ; ЧАСТОТОЙ 100 ГЦ, БУДЕТ 4000000/100 = 40000.

        ;ПРОИЗВОЛЬНЫЕ АДРЕСА ПОРТОВ 8253
PIT0    EQU     0B0H            ;СЧЕТЧИК 0 8253
PIT1    EQU     0B1H            ;СЧЕТЧИК 1 8253
PIT2    EQU     0B2H            ;СЧЕТЧИК 2 8253
PITMDE  EQU     0B3H            ;РЕГИСТР УПРАВЛЯЮЩЕГО СЛОВА 8253

        ;БАЙТ РЕЖИМА РАБОТЫ И ЗНАЧЕНИЕ СЧЕТЧИКА 8253
PITCTRL EQU     00110110B       ;РАЗРЯД 0 = 0 (ДВОИЧНЫЙ РЕЖИМ РАБОТЫ)
                                ;РАЗРЯДЫ 3..1 = 011 (РЕЖИМ 3 - ГЕНЕРАТОР
                                ; ПОСЛЕДОВАТЕЛЬНОСТИ ПРЯМОУГОЛЬНЫХ
                                ; ИМПУЛЬСОВ)
                                ;РАЗРЯДЫ 5,4 = 11 (ЗАГРУЗИТЬ В СЧЕТЧИК
                                ; 2 БАЙТА)
                                ;РАЗРЯДЫ 7,6 = 00 (ПРОГРАММНЫЙ
                                ; СЧЕТЧИК 0)
PITCNT  EQU     40000           ;ЗНАЧЕНИЕ СЧЕТЧИКА = 40000

        ;ЗНАЧЕНИЕ ИНТЕРВАЛА ПО УМОЛЧАНИЮ (100 ГЦ ДЛЯ ЧАСОВ РЕАЛЬНОГО
        ; ВРЕМЕНИ)
DTICK   EQU     100             ;ЗНАЧЕНИЕ ИНТЕРВАЛА ПО УМОЛЧАНИЮ

        ;ЭКВИВАЛЕНТНЫЕ ЗНАЧЕНИЯ ДЛЯ ПРОГРАММИРУЕМОГО КОНТРОЛЛЕРА
        ; ПРЕРЫВАНИЙ (PIC) 8259.
        ; 8259 ПРОГРАММИРУЕТСЯ ДЛЯ СЛЕДУЮЩЕГО РЕЖИМА РАБОТЫ:
        ;  ОДНО УСТРОЙСТВО (В ОТЛИЧИЕ ОТ НЕСКОЛЬКИХ КОНТРОЛЛЕРОВ 8259),
        ;  РЕЖИМ ПОЛНОСТЬЮ ВЛОЖЕННЫХ ПРЕРЫВАНИЙ,
        ;  ВСЕ ПРЕРЫВАНИЯ РАЗРЕШЕНЫ,
        ;  RESTART 3, АДРЕС 0018, ПРЕРЫВАНИЯ ПО ВРЕМЕНИ.
        ;ПРОИЗВОЛЬНЫЕ АДРЕСА ПОРТОВ 8259
PIC0    EQU     0B4H            ;ПОРТ 1 PIC
PIC1    EQU     0B5H            ;ПОРТ 2 PIC

        ;ВЕКТОР ПРЕРЫВАНИЙ
CLKITRP EQU     0018H           ;ВЕКТОР ПРЕРЫВАНИЙ ПО ВРЕМЕНИ

        ;КОМАНДНЫЕ СЛОВА ICW1 И ICW2 ДЛЯ ИНИЦИАЛИЗАЦИИ 8259
ICW1    EQU     00010010B       ;РАЗРЯД 0 = 0 (НЕ ИСПОЛЬЗУЕТСЯ)
                                ;РАЗРЯД 1 = 1 (ОДИН КОНТРОЛЛЕР 8259)
                                ;РАЗРЯД 2 = 0 (ИНТЕРВАЛ ВЕКТОРА = 8 БАЙТ)
                                ;РАЗРЯД 3 = 0 (НЕ ИСПОЛЬЗУЕТСЯ)
                                ;РАЗРЯД 4 = 1 (ДОЛЖЕН БЫТЬ РАВЕН 1)
                                ;РАЗРЯДЫ 7,6,5 = 0 (РАЗРЯДЫ 5-7 БАЗОВОГО
```

```
                                        ; АДРЕСА ДЛЯ КОМАНД RST)
ICW2    EQU     0                       ;СТАРШИЙ БАЙТ БАЗОВОГО АДРЕСА ДЛЯ КОМАНД
                                        ; RST)

        ;КОМАНДНОЕ СЛОВО ДЛЯ РАБОТЫ 8259
EOI     EQU     00100000B               ;КОНЕЦ КОМАНДНОГО СЛОВА ПРЕРЫВАНИЯ

        ;ВЕРНУТЬ БАЗОВЫЙ АДРЕС ПЕРЕМЕННЫХ CLOCK
CLOCK:
        LXI     H,CLKVAR                ;ВЗЯТЬ БАЗОВЫЙ АДРЕС ПЕРЕМЕННЫХ CLOCK
        RET

        ;ИНИЦИАЛИЗИРОВАТЬ СЧЕТЧИК 0 8253 ДЛЯ ПРЕРЫВАНИЙ ПО ИНТЕРВАЛАМ
        ; ВРЕМЕНИ
ICLK:
        DI                              ;ЗАПРЕТИТЬ ПРЕРЫВАНИЯ
        MVI     A,0C3H                  ;ШЕСТНАДЦАТЕРИЧНОЕ ЧИСЛО C3 - ЭТО КОД
                                        ; ОПЕРАЦИИ КОМАНДЫ JMP
        STA     CLKITRP                 ;ЗАПОМНИТЬ КОД ОПЕРАЦИИ КОМАНДЫ JMP
                                        ; В КАЧЕСТВЕ АДРЕСА ДЛЯ ПРЕРЫВАНИЯ
        LXI     H,CLKINT
        SHLD    CLKITRP+1               ;ЗАПОМНИТЬ АДРЕС ПЕРЕХОДА

        ;ИНИЦИАЛИЗИРОВАТЬ КОНТРОЛЛЕР ПРЕРЫВАНИЙ 8259
        MVI     A,ICW1
        OUT     PIC0                    ;ПОСЛАТЬ ПЕРВОЕ СЛОВО В ПОРТ 0
        MVI     A,ICW2
        OUT     PIC1                    ;ПОСЛАТЬ ВТОРОЕ СЛОВО В ПОРТ 1

        ;ИНИЦИАЛИЗИРОВАТЬ ПРОГРАММИРУЕМЫЙ ИНТЕРВАЛЬНЫЙ ТАЙМЕР 8253
        MVI     A,PITCTRL               ;ВЫВЕСТИ УПРАВЛЯЮЩЕЕ СЛОВО
        OUT     PITMDE
        LXI     H,PITCNT                ;ВЫВЕСТИ ДВА БАЙТА НАЧАЛЬНОГО
        MOV     A,L                     ; СЧЕТЧИКА
        OUT     PIT0
        MOV     A,H
        OUT     PIT0

        ;НАЧАЛЬНЫЕ ПРОИЗВОЛЬНЫЕ ЗНАЧЕНИЯ ПЕРЕМЕННЫХ CLOCK
        ;1 ЯНВАРЯ 1980 ГОДА 00:00.00
        ;ДЛЯ РАБОТЫ РЕАЛЬНЫХ ЧАСОВ ТРЕБУЕТСЯ ИЗМЕНИТЬ ЭТИ ЗНАЧЕНИЯ
        LXI     H,TICK
        MVI     M,DTICK                 ;ИНИЦИАЛИЗИРОВАТЬ ЗНАЧЕНИЕ ИНТЕРВАЛОВ
        INX     H
        SUB     A                       ;A = 0
        MOV     M,A                     ;0 СЕКУНД
        INX     H
        MOV     M,A                     ;0 МИНУТ
        INX     H
        MOV     M,A                     ;0 ЧАСОВ
        INR     A                       ;A = 1
        INX     H
        MOV     M,A                     ;1-Й ДЕНЬ
        INX     H
        MOV     M,A                     ;1-Й МЕСЯЦ (ЯНВАРЬ)
        LXI     H,1980
        SHLD    YEAR                    ;1980 ГОД
```

```
        EI
        RET

        ;ОБСЛУЖИТЬ ПРЕРЫВАНИЕ ПО ВРЕМЕНИ
CLKINT:
        PUSH    PSW             ;СОХРАНИТЬ AF, HL
        PUSH    H
        LXI     H,TICK          ;ВЫЧЕСТЬ 1 ИЗ СЧЕТЧИКА ИНТЕРВАЛОВ
        DCR     M
        JNZ     EXIT1           ;ПЕРЕЙТИ, ЕСЛИ СЧЕТЧИК ИНТЕРВАЛОВ
                                ; НЕ РАВЕН НУЛЮ
        MVI     M,DTICK         ;ОПЯТЬ УСТАНОВИТЬ ЗНАЧЕНИЕ СЧЕТЧИКА
                                ; РАВНЫМ ЗНАЧЕНИЮ ПО УМОЛЧАНИЮ

        ;СОХРАНИТЬ ОСТАЛЬНЫЕ РЕГИСТРЫ
        PUSH    B               ;СОХРАНИТЬ BC, DE
        PUSH    D

        MVI     B,0             ;0 = НАЧАЛЬНЫЕ ЗНАЧЕНИЯ ПО УМОЛЧАНИЮ
                                ; ДЛЯ СЕКУНД, МИНУТ, ЧАСОВ

        ;ПРИБАВИТЬ СЕКУНДУ
        INX     H               ;УСТАНОВИТЬ УКАЗАТЕЛЬ НА ЧИСЛО СЕКУНД
        INR     M               ;ПРИБАВИТЬ СЕКУНДУ
        MOV     A,M
        CPI     60              ;60 СЕКУНД?
        JC      EXIT0           ;ВЫЙТИ, ЕСЛИ МЕНЬШЕ, ЧЕМ 60 СЕКУНД
        MOV     M,B             ;ИНАЧЕ 0 СЕКУНД

        ;ПРИБАВИТЬ МИНУТУ
        INX     H               ;УСТАНОВИТЬ УКАЗАТЕЛЬ НА ЧИСЛО МИНУТ
        INR     M               ;ПРИБАВИТЬ МИНУТУ
        MOV     A,M
        CPI     60              ;60 МИНУТ?
        JC      EXIT0           ;ВЫЙТИ, ЕСЛИ МЕНЬШЕ, ЧЕМ 60 МИНУТ
        MOV     M,B             ;ИНАЧЕ 0 МИНУТ

        ;ПРИБАВИТЬ ЧАС
        INX     H               ;УСТАНОВИТЬ УКАЗАТЕЛЬ НА ЧИСЛО ЧАСОВ
        INR     M               ;ПРИБАВИТЬ ЧАС
        MOV     A,M
        CPI     24              ;24 ЧАСА?
        JC      EXIT0           ;ВЫЙТИ, ЕСЛИ МЕНЬШЕ, ЧЕМ 24 ЧАСА
        MOV     M,B             ;ИНАЧЕ 0 ЧАСОВ

        ;ПРИБАВИТЬ ДЕНЬ
        XCHG                    ;DE = АДРЕС ЧИСЛА ЧАСОВ
        LXI     H,LASTDY-1
        LDA     MONTH           ;ВЗЯТЬ НОМЕР ТЕКУЩЕГО МЕСЯЦА
        MOV     C,A             ;РЕГИСТР C = НОМЕР МЕСЯЦА
        MVI     B,0
        DAD     B               ;УСТАНОВИТЬ УКАЗАТЕЛЬ НА НОМЕР
                                ; ПОСЛЕДНЕГО ДНЯ МЕСЯЦА
        XCHG                    ;HL = АДРЕС ЧИСЛА ЧАСОВ
        INX     H               ;УСТАНОВИТЬ УКАЗАТЕЛЬ НА НОМЕР ДНЯ
        MOV     A,M             ;ВЗЯТЬ НОМЕР ДНЯ
        INR     M               ;ПРИБАВИТЬ ДЕНЬ
        XCHG                    ;DE = АДРЕС НОМЕРА ДНЯ
```

```
        MOV     B,A                     ;РЕГИСТР B = НОМЕР ДНЯ
        CMP     M                       ;ТЕКУЩИЙ ДЕНЬ - ПОСЛЕДНИЙ ДЕНЬ МЕСЯЦА?
        XCHG                            ;HL = АДРЕС НОМЕРА ДНЯ
        JC      EXITO                   ;ВЫЙТИ, ЕСЛИ НЕ КОНЕЦ МЕСЯЦА

        ;ОПРЕДЕЛИТЬ, НИ КОНЕЦ ЛИ ФЕВРАЛЯ В ВИСОКОСНОМ ГОДУ
        ; (КОГДА ГОД ДЕЛИТСЯ НА 4)?
        MOV     A,C                     ;ВЗЯТЬ НОМЕР МЕСЯЦА
        CPI     2                       ;ЭТО ФЕВРАЛЬ?
        JNZ     INCMTH                  ;ПЕРЕЙТИ, ЕСЛИ НЕТ, ПРИБАВИТЬ МЕСЯЦ
        LDA     YEAR                    ;ГОД ВИСОКОСНЫЙ?
        ANI     00000011B
        JNZ     INCMTH                  ;ПЕРЕЙТИ, ЕСЛИ НЕТ

        ;В ФЕВРАЛЕ ВИСОКОСНОГО ГОДА 29 ДНЕЙ, А НЕ 28
        MOV     A,B                     ;ВЗЯТЬ НОМЕР ДНЯ
        CPI     29
        JC      EXITO                   ;ВЫЙТИ, ЕСЛИ НЕ 1-Е МАРТА

INCMTH:
        MVI     B,1                     ;НОМЕРА ДНЯ И МЕСЯЦА ПО УМОЛЧАНИЮ РАВНЫ 1
        MOV     M,B                     ;1-Й ДЕНЬ

        INX     H
        INR     M                       ;ПРИБАВИТЬ МЕСЯЦ
        MOV     A,C                     ;ВЗЯТЬ НОМЕР СТАРОГО МЕСЯЦА
        CPI     12                      ;БЫЛ ДЕКАБРЬ?
        JC      EXITO                   ;ВЫЙТИ, ЕСЛИ НЕТ
        MOV     M,B                     ;ИНАЧЕ
                                        ; ИЗМЕНИТЬ НОМЕР МЕСЯЦА НА 1-Й (ЯНВАРЬ)
        ;ПЕРЕЙТИ НА СЛЕДУЮЩИЙ ГОД
        LHLD    YEAR
        INX     H
        SHLD    YEAR

EXITO:
        ;ВОССТАНОВИТЬ РЕГИСТРЫ
        POP     D                       ;ВОССТАНОВИТЬ DE, BC
        POP     B
EXIT1:
        POP     H                       ;ВОССТАНОВИТЬ HL
        MVI     A,EOI                   ;ОЧИСТИТЬ ПРЕРЫВАНИЯ 8259
        OUT     PICO
        POP     PSW                     ;ВОССТАНОВИТЬ AF
        EI                              ;ВНОВЬ РАЗРЕШИТЬ ПРЕРЫВАНИЯ
        RET                             ;ВОЗВРАТИТЬСЯ

        ;МАССИВ НОМЕРОВ ПОСЛЕДНИХ ДНЕЙ ДЛЯ КАЖДОГО МЕСЯЦА
LASTDY:
        DB      31                      ;ЯНВАРЬ
        DB      28                      ;ФЕВРАЛЬ (ЗА ИСКЛЮЧЕНИЕМ ВИСОКОСНЫХ
                                        ; ЛЕТ)
        DB      31                      ;МАРТ
        DB      30                      ;АПРЕЛЬ
        DB      31                      ;МАЙ
        DB      30                      ;ИЮНЬ
        DB      31                      ;ИЮЛЬ
```

```
        DB      31                  ;АВГУСТ
        DB      30                  ;СЕНТЯБРЬ
        DB      31                  ;ОКТЯБРЬ
        DB      30                  ;НОЯБРЬ
        DB      31                  ;ДЕКАБРЬ

        ;ПЕРЕМЕННЫЕ CLOCK
CLKVAR:
TICK:   DS      1                   ;ЧИСЛО ИНТЕРВАЛОВ, ОСТАВШИХСЯ
                                    ; В ТЕКУЩЕЙ СЕКУНДЕ
SEC:    DS      1                   ;ЧИСЛО СЕКУНД
MIN:    DS      1                   ;ЧИСЛО МИНУТ
HOUR:   DB      1                   ;ЧИСЛО ЧАСОВ
DAY:    DB      1                   ;НОМЕР ДНЯ (1 ДЛЯ ЧИСЛА ДНЕЙ В МЕСЯЦЕ)
MONTH:  DB      1                   ;НОМЕР МЕСЯЦА, 1 = ЯНВАРЬ .. 12 = ДЕКАБРЬ
YEAR:   DB      2                   ;ГОД


;                                                                       ;
;                                                                       ;
;       ПРИМЕР ВЫПОЛНЕНИЯ                                               ;
;                                                                       ;
;                                                                       ;

        ;ИНДЕКСЫ ДЛЯ ПЕРЕМЕННЫХ CLOCK
TCKIDX  EQU     0                   ;ИНДЕКС ДЛЯ ЧИСЛА ИНТЕРВАЛОВ
SECIDX  EQU     1                   ;ИНДЕКС ДЛЯ ЧИСЛА СЕКУНД
MINIDX  EQU     2                   ;ИНДЕКС ДЛЯ ЧИСЛА МИНУТ
HRIDX   EQU     3                   ;ИНДЕКС ДЛЯ ЧИСЛА ЧАСОВ
DAYIDX  EQU     4                   ;ИНДЕКС ДЛЯ НОМЕРА ДНЯ
MTHIDX  EQU     5                   ;ИНДЕКС ДЛЯ НОМЕРА МЕСЯЦА
YRIDX   EQU     6                   ;ИНДЕКС ДЛЯ ГОДА

SC11D:
        CALL    ICLK                ;ИНИЦИАЛИЗИРОВАТЬ ЧАСЫ

        ;ИНИЦИАЛИЗИРОВАТЬ ЧАСЫ, ЗАДАВ НАЧАЛЬНОЕ ЗНАЧЕНИЕ
        ; 2/7/83 14:00:00 (2 ЧАСА ДНЯ, 7 ФЕВРАЛЯ, 1983 ГОД)
        CALL CLOCK                  ;HL = АДРЕС ПЕРЕМЕННЫХ CLOCK
        DI                          ;ЗАПРЕТИТЬ ПРЕРЫВАНИЯ НА ВРЕМЯ
                                    ;ИНИЦИАЛИЗАЦИИ ЧАСОВ
        INX     H                   ;ПРОПУСТИТЬ ЧИСЛО ИНТЕРВАЛОВ
        MVI     M,0                 ;0 СЕКУНД
        INX     H
        MVI     M,0                 ;0 МИНУТ
        INX     H
        MVI     M,14                ;14 ЧАСОВ (2 ЧАСА ДНЯ)
        INX     H
        MVI     M,7                 ;7-Й ДЕНЬ
        INX     H
        MVI     M,2                 ;2-Й МЕСЯЦ (ФЕВРАЛЬ)
        LXI     D,1983
        INX     H
        MOV     M,E                 ;1983 ГОД
        INX     H
        MOV     M,D
        EI                          ;ВНОВЬ РАЗРЕШИТЬ ПРЕРЫВАНИЯ
```

```
        ;ЖДАТЬ, ПОКА НЕ НАСТАНЕТ 2/7/83 14:01:20
        ; (2:01.20 ДНЯ, 7 ФЕВРАЛЯ, 1983 ГОД)
        ;
        ;ПРИМЕЧАНИЕ: НЕОБХОДИМО ПРАВИЛЬНО ОРГАНИЗОВАТЬ ВЫХОД ПРИ ОЖИДАНИИ
        ; НАСТУПЛЕНИЯ СОБЫТИЯ, УЧИТЫВАЯ ВОЗМОЖНОСТЬ ТОГО, ЧТО В МОМЕНТ
        ; СРАВНЕНИЯ ЧАСЫ МОГУТ УЖЕ УЙТИ ВПЕРЕД. ПРИ ПРОВЕРКЕ ТОЛЬКО НА
        ; РАВЕНСТВО  МОЖНО НИКОГДА НЕ ПОЛУЧИТЬ ЕГО. ПОЭТОМУ В ПРОВЕРКАХ,
        ; ПРИВЕДЕННЫХ НИЖЕ, МЫ ИМЕЕМ >=, А НЕ =.

        ;ЖДАТЬ, ПОКА ГОД БУДЕТ >= 1983
        CALL CLOCK              ;HL = БАЗОВЫЙ АДРЕС ПЕРЕМЕННЫХ CLOCK
        PUSH    H               ;СОХРАНИТЬ БАЗОВЫЙ АДРЕС
        LXI     D,YRIDX
        DAD     D               ;HL = АДРЕС МЛАДШЕГО БАЙТА ГОДА
        LXI     D,1983          ;DE = ОЖИДАЕМЫЙ ГОД
WAITYR:
        ;ПОЛУЧИТЬ ТЕКУЩИЙ ГОД ПРИ ЗАПРЕЩЕННЫХ ПРЕРЫВАНИЯХ, ТАК КАК ГОД
        ; ХРАНИТСЯ В ДВУХ БАЙТАХ
        DI                      ;ЗАПРЕТИТЬ ПРЕРЫВАНИЯ НА ВРЕМЯ ВЫДЕЛЕНИЯ
                                ; 2-БАЙТНОГО ГОДА
        MOV     C,M             ;ВЗЯТЬ МЛАДШИЙ БАЙТ ГОДА
        INX     H
        MOV     B,M             ;ВЗЯТЬ СТАРШИЙ БАЙТ ГОДА
        DCX     H
        EI                      ;ВНОВЬ РАЗРЕШИТЬ ПРЕРЫВАНИЯ

        ;СРАВНИТЬ ТЕКУЩИЙ ГОД И 1983
        MOV     A,C
        CMP     E
        MOV     A,B
        SBB     D
        JC      WAITYR          ;ПЕРЕЙТИ, ЕСЛИ ГОД НЕ >= 1983

        ;ЖДАТЬ, ПОКА НОМЕР МЕСЯЦА НЕ БУДЕТ >= 2
        POP     H               ;HL = БАЗОВЫЙ АДРЕС ПЕРЕМЕННЫХ CLOCK
        LXI     D,MTHIDX
        DAD     D               ;УСТАНОВИТЬ УКАЗАТЕЛЬ НА НОМЕР МЕСЯЦА
        MVI     B,2
        CALL    WAIT            ;ЖДАТЬ ФЕВРАЛЯ ИЛИ СЛЕДУЮЩЕГО ЗА НИМ
                                ; МЕСЯЦА

        ;ЖДАТЬ, ПОКА НОМЕР ДНЯ НЕ  БУДЕТ >= 7
        DCX     H               ;УСТАНОВИТЬ УКАЗАТЕЛЬ НА НОМЕР ДНЯ
        MVI     B,7
        CALL    WAIT            ;ЖДАТЬ, ПОКА НЕ БУДЕТ 7-Й ДЕНЬ ИЛИ ПОЗЖЕ

        ;ЖДАТЬ, ПОКА ЧИСЛО ЧАСОВ НЕ БУДЕТ >= 14
        DCX     H               ;УСТАНОВИТЬ УКАЗАТЕЛЬ НА ЧИСЛО ЧАСОВ
        MVI     B,14
        CALL    WAIT            ;ЖДАТЬ, ПОКА НЕ БУДЕТ 2 ЧАСА ДНЯ
                                ; ИЛИ ПОЗЖЕ

        ;ЖДАТЬ, ПОКА ЧИСЛО МИНУТ НЕ БУДЕТ >= 1
        DCX     H               ;УСТАНОВИТЬ УКАЗАТЕЛЬ НА ЧИСЛО МИНУТ
        MVI     B,1
        CALL    WAIT            ;ЖДАТЬ, ПОКА НЕ БУДЕТ 2:01 ИЛИ ПОЗЖЕ
```

```
        ;ЖДАТЬ, ПОКА ЧИСЛО СЕКУНД НЕ БУДЕТ >= 20
        DCX     H                   ;УСТАНОВИТЬ УКАЗАТЕЛЬ НА ЧИСЛО СЕКУНД
        MVI     B,20
        CALL    WAIT                ;ЖДАТЬ, ПОКА НЕ БУДЕТ 2:01.20 ИЛИ ПОЗЖЕ

        ;ДОЖДАЛИСЬ
HERE:
        JMP     HERE                ;СЕЙЧАС ЗАДАННОЕ ВРЕМЯ ИЛИ ПОЗЖЕ

        ;**********************************
        ;ПОДПРОГРАММА: WAIT
        ;НАЗНАЧЕНИЕ:   ЖДАТЬ, ПОКА ЗНАЧЕНИЕ, АДРЕС КОТОРОГО ЗАДАН В HL,
        ;              НЕ БУДЕТ БОЛЬШЕ ИЛИ РАВНО ЗНАЧЕНИЮ В РЕГИСТРЕ B
        ;ВХОД:  HL = АДРЕС ПЕРЕМЕННОЙ, ЗА КОТОРОЙ НАДО СЛЕДИТЬ
        ;       B = ОЖИДАЕМОЕ ЗНАЧЕНИЕ
        ;ВЫХОД: КОГДА B >= (HL)
        ;ИСПОЛЬЗУЕМЫЕ РЕГИСТРЫ: AF
        ;**********************************
WAIT:
        MOV     A,M                 ;ВЗЯТЬ ЧАСТЬ ИЗ ПЕРЕМЕННЫХ CLOCK
        CMP     B                   ;СРАВНИТЬ С ЗАДАННЫМ ЗНАЧЕНИЕМ
        JC      WAIT                ;ЖДАТЬ, ПОКА ОНО НЕ БУДЕТ ДОСТИГНУТО
        RET

        END
```

ПРИЛОЖЕНИЕ А

СИСТЕМА КОМАНД МИКРОПРОЦЕССОРОВ 8080, 8085

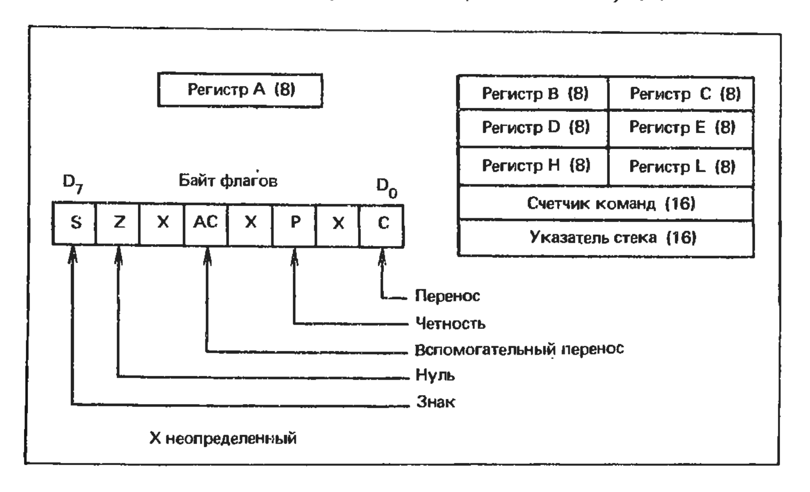


Рис. А.1. Внутренняя организация регистров микропроцессоров 8080, 8085

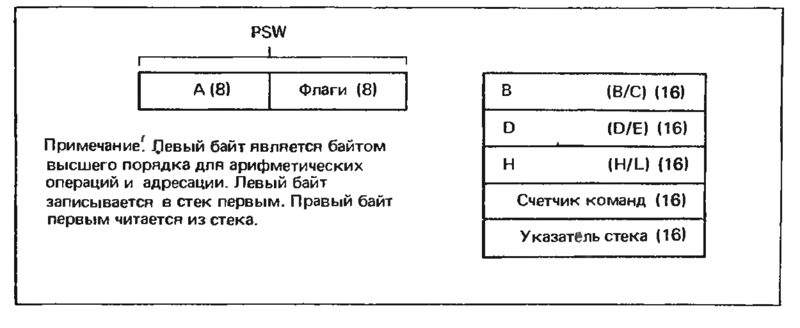


Рис. А.2. Организация пар регистров микропроцессоров 8080, 8085

Использование регистра А при выполнении команд RIM и SIM (только в 8085)

Регистр А после выполнения команды RIM

D_7 … D_0

SID	I7.5	I6.5	I5.5	IE	M7.5	M6.5	M5.5

- M7.5, M6.5, M5.5 — Маски прерываний
- IE — Флаг разрешений прерываний
- I7.5, I6.5, I5.5 — Отложенные прерывания
- SID — Данные последовательного ввода

Регистр А перед выполнением команды SIM

D_7 … D_0

SOD	SOE	X	R7.5	MSE	M7.5	M6.5	M5.5

Не определено

- M5.5 — Маска RST 5.5
- M6.5 — Маска RST 6.5
- M7.5 — Маска RST 7.5
- MSE — Установка масок разрешена
- R7.5 — Сброс RST 7.5
- X — Не опраделено
- SOE — Последовательный вывод (SOD) разрешен
- SOD — Данные последовательного вывода

Рис. А.3. Команды RIM и SIM 8085

Таблица А.1. *Команды микропроцессоров 8080, 8085 в алфавитном порядке*

Команда[1]		Объектный код	Число байтов	Число тактов	
				8085	8080
ACI	DATA	CE YY	2	7	7
ADC	REG	10001XXX	1	4	4
ADC	M	8E	1	7	7
ADD	REG	10000XXX	1	4	4
ADD	M	86	1	7	7
ADI	DATA	C6 YY	2	7	7
ANA	REG	10100XXX	1	4	4
ANA	M	A6	I	7	7
ANI	DATA	E6 YY	2	7	7
CALL	LABEL	CD ppqq	3	18	17
CC	LABEL	DC ppqq	3	9/18	11/17
CM	LABEL	FC ppqq	3	9/18	11/17
CMA		2F	1	4	4
CMC		3F	1	4	4
CMP	REG	10111XXX	1	4	4
CMP	M	BE	1	7	7
CNC	LABEL	D4 ppqq	3	9/18	11/17
CNZ	LABEL	C4 ppqq	3	9/18	11/17
CP	LABEL	F4 ppqq	3	9/18	11/17
CPE	LABEL	EC ppqq	3	9/18	11/17
CPI	DATA	FE YY	2	7	7
CPO	LABEL	E4 ppqq	3	9/18	11/17
CZ	LABEL	CC ppqq	3	9/18	11/17
DAA		27	1	4	4
DAD	RP	00XX1001	1	10	10
DCR	REG	00XXX101	1	4	5
DCR	M	35	1	10	10
DCX	RP	00XX1011	1	6	5
DI		F3	1	4	4
EI		FB	1	4	4
HLT		76	1	4	4
IN	PORT	DB YY	2	10	10
INR	REG	00XXX100	1	4	5
INR	M	34	1	10	10
INX	RP	00XX0011	1	6	5
JC	LABEL	DA ppqq	3	7/10	10
JM	LABEL	FA ppqq	3	7/10	10
JMP	LABEL	C3 ppqq	3	10	10
JNC	LABEL	D2 ppqq	3	7/10	10
JNZ	LABEL	C2 ppqq	3	7/10	10
JP	LABEL	F2 ppqq	3	7/10	10
JPE	LABEL	EA ppqq	3	7/10	10
JPO	LABEL	E2 ppqq	3	7/10	10
JZ	LABEL	CA ppqq	3	7/10	10

Команда[1]		Объектный код	Число байтов	Число тактов	
				8085	8080
LDA	ADDR	3A ppqq	3	13	13
LDAX	RP	000X1010	1	7	7
LHLD	ADDR	2A ppqq	3	16	16
LXI	RP, DATA 16	00XX0001 YYYY	3	10	10
MOV	REG, REG	01dddsss	1	4	5
MOV	M, REG	01110 sss	1	7	7
MOV	REG, M	01ddd110	1	7	7
MVI	REG, DATA	00ddd110 YY	2	7	7
MVI	M, DATA	36 YY	2	10	10
NOP		00	1	4	4
ORA	REG	10110XXX	1	4	5
ORA	M	B6	1	7	7
ORI	DATA	F6 YY	2	7	7
OUT	PORT	D3 YY	2	10	10
PCHL		E9	1	6	5
POP	PR	11XX0001	1	10	10
PUSH	RP	11XX0101	1	12	11
RAL		17	1	4	4
RAR		1F	1	4	4
RC		D8	1	6/12	5/11
RET		C9	1	10	10
RIM*		20	1	4	
RLC		07	1	4	4
RM		F8	1	6/12	5/11
RNC		D0	1	6/12	5/11
RNZ		C0	1	6/12	5/11
RP		F0	1	6/12	5/11
RPE		E8	1	6/12	5/11
RPO		E0	1	6/12	5/11
RRC		0F	1	4	4
RST	N	11nnn111	1	12	11
RZ		C8	1	6/12	5/11
SBB	REG	10011XXX	1	4	4
SBB	M	9E	1	7	7
SBI	DATA	DE YY	2	7	7
SHLD	ADDR	22 ppqq	3	16	16
SIM*		30	1	4	
SPHL		F9	1	6	5
STA	ADDR	32 ppqq	3	13	13
STAX	RP	000X0010	1	7	7
STC		37	1	4	4
SUB	REG	10010XXX	1	4	4
SUB	M	96	1	7	7
SUI	DATA	D6 YY	2	7	7
XCHG		EB	1	4	4

Команда[1]		Объектный код	Число байтов	Число тактов	
				8085	8080
XRA	REG	10101XXX	1	4	4
XRA	M	AE	1	7	7
XRI	DATA	EE YY	2	7	7
XTHL		E3	1	16	18

В объектном коде ddd – регистр назначения; кодируется так же, как XXX; nnn – номер рестарта от 000 до 111; ppqq – 16-разрядный адрес памяти; sss – регистр-источник; кодируется так же, как XXX; X – пара регистров 0 = BC, 1 = DE; XX – пара регистров 00 = BC, 01 = DE, 10 = HL, 11 = SP или (если PUSH/POP) PSW; XXX – регистр 111 = A, 000 = B, 001 = C, 010 = D, 011 = E, 100 = H, 101 = L; YY – 8-разрядные двоичные данные; YYYY – 16-разрядные двоичные данные.

* Команды только 8085.

[1] Для операндов приняты следующие обозначения: ADDR – адрес, DATA – 8-разрядные данные; DATA16 – 16-разрядные данные, LABEL – метка, PORT – порт, REG – регистр, RP – пара регистров. *Прим. перев.*

Таблица А.2. *Система команд микропроцессоров 8080, 8085*

Группа команд передачи данных

Переслать

MOV	A,A	7F	MOV	C,A	4F	MOV	E,A	5F	MOV	L,A	6F
	A,B	78		C,B	48		E,B	58		L,B	68
	A,C	79		C,C	49		E,C	59		L,C	69
	A,D	7A		C,D	4A		E,D	5A		L,D	6A
	A,E	7B		C,E	4B		E,E	5B		L,E	6B
	A,H	7C		C,H	4C		E,H	5C		L,H	6C
	A,L	7D		C,L	4D		E,L	5D		L,L	6D
	A,M	7E		C,M	4E		E,M	5E		L,M	6E
MOV	B,A	47	MOV	D,A	57	MOV	H,A	67	MOV	M,A	77
	B,B	40		D,B	50		H,B	60		M,B	70
	B,C	41		D,C	51		H,C	61		M,C	71
	B,D	42		D,D	52		H,D	62		M,D	72
	B,E	43		D,E	53		H,E	63		M,E	73
	B,H	44		D,H	54		H,H	64		M,H	74
	B,L	45		D,L	55		H,L	65		M,L	75
	B,M	46		D,M	56		H,M	66	XCHG		EB

Переслать непосредственно			Загрузить непосредственно			Загрузить, запомнить	
MVI	A,byte	3E	LXI	B,dble	01	LDAX B	0A
	B,byte	06		D,dble	11	LDAX D	1A
	C,byte	0E		H,dble	21	LHLD adr	2A
	D,byte	16		SP,dble	31	LDA adr	3A
	E,byte	1E				STAX B	02
	H,byte	26				STAX D	12
	L,byte	2E				SHLD adr	22
	M,byte	36				STA adr	32

Группа арифметических и логических команд

Сложить[1]

ADD	A	87
	B	80
	C	81
	D	82
	E	83
	H	84
	L	85
	M	86
ADC	A	8F
	B	83
	C	89
	D	8A
	E	8B
	H	8C
	L	8D
	M	8E

Вычесть[1]

SUB	A	97
	B	90
	C	91
	D	92
	E	93
	H	94
	L	95
	M	96
SBB	A	9F
	B	98
	C	99
	D	9A
	E	9B
	H	9C
	L	9D
	M	9E

Сложить пары регистров[3]

DAD	B	09
	D	19
	H	29
	SP	39

Увеличить[2]

INR	A	3C
	B	04
	C	0C
	D	14
	E	1C
	H	24
	L	2C
	M	34
INX	B	03
	D	13
	H	23
	SP	33

Уменьшить[2]

DCR	A	3D
	B	05
	C	0D
	D	15
	E	1D
	H	25
	L	2D
	M	35
DCX	B	0B
	D	1B
	H	2B
	SP	3B

Специальные

DAA[1]	27
CMA	2F
STC[3]	37
CMC[3]	3F

Логические[1]

ANA	A	A7
	B	A0
	C	A1
	D	A2
	E	A3
	H	A4
	L	A5
	M	A6
XRA	A	AF
	B	A8
	C	A9
	D	AA
	E	AB
	H	AC
	L	AD
	M	AE
ORA	A	B7
	B	B0
	C	B1
	D	B2
	E	B3
	H	B4
	L	B5
	M	B6
CMP	A	BF
	B	B8
	C	B9
	D	BA
	E	BB
	H	BC
	L	BD
	M	BE

Сдвинуть циклически[3]

RLC	07
RRC	0F
RAL	17
RAR	1F

Арифметические и логические непосредственные

ADI	byte	C6
ACI	byte	CE
SUI	byte	D6
SBI	byte	DE
ANI	byte	E6
XRI	byte	EE
ORI	byte	F6
CPI	byte	FE

Группа команд передачи управления

Перейти			Вызвать			Возвратиться		Рестарт		
JMP	adr	C3	CALL	adr	CD	RET	C9	RST	0	C7
JNZ	adr	C2	CNZ	adr	C4	RNZ	C0		1	CF
JZ	adr	CA	CZ	adr	CC	RZ	C8		2	D7
JNC	adr	D2	CNC	adr	D4	RNC	D0		3	DF
JC	adr	DA	CC	adr	DC	RC	D8		4	E7
JPO	adr	E2	CPO	adr	E4	RPO	E0		5	EF
JPE	adr	EA	CPE	adr	EC	RPE	E8		6	F7
JP	adr	F2	CP	adr	F4	RP	F0		7	FF
JM	adr	FA	CM	adr	FC	RM	F8			
PCHL	adr	E9								

Команды ввода-вывода и управления ЭВМ

Операции со стеком			Ввод-вывод			Управление		Новые команды (только в 8085)	
PUSH	B	C5	OUT	byte	D3	DI	F3	RIM	20
	D	D5	IN	byte	DB	EI	FB	SIM	30
	H	E5				NOP	00		
	PSW	F5				HLT	76		
POP	B	C1							
	D	D1							
	H	E1							
	PSW[1]	F1							
XTHL	E3								
SPHL	F9								

byte — константа или арифметическое или логическое выражение, результатом вычисления которого являются 8-разрядные данные (второй байт двухбайтных команд); dble — константа или арифметическое или логическое выражение, результатом вычисления которого являются 16-разрядные данные (второй и третий байты трехбайтных команд); adr — 16-разрядный адрес (второй и третий байты трехбайтных команд).

[1] Команды влияют на все флаги (C, Z, S, P, AC).

[2] Команды влияют на все флаги за исключением флага переноса (исключением являются команды INX и DCX, которые не влияют на флаги).

[3] Команды влияют только на флаг переноса.

Таблица А.3. *Коды операций микропроцессоров 8080, 8085 в порядке возрастания их числовых значений*

00	NOP		0E	MVI	C,byte	1C	INR	E
01	LXI	B,dble	0F	RRC		1D	DCR	E
02	STAX	B	10	–		1E	MVI	E,byte
03	INX	B	11	LXI	D,dble	1F	RAR	
04	INR	B	12	STAX	D	20	RIM[1]	
05	DCR	B	13	INX	D	21	LXI	H,dble
06	MVI	B,byte	14	INR	D	22	SHLD	adr
07	RLC		15	DCR	D	23	INX	H
08	–		16	MVI	D,byte	24	INR	H
09	DAD	B	17	RAL		25	DCR	H
0A	LDAX	B	18	–		26	MVI	H,byte
0B	DCX	B	19	DAD	D	27	DAA	
0C	INR	C	1A	LDAX	D	28	–	
0D	DCR	C	1B	DCX	D	29	DAD	H

2A	LHLD	adr	5D	MOV	E,L	90	SUB	B
2B	DCX	H	5E	MOV	E,M	91	SUB	C
2C	INR	L	5F	MOV	E,A	92	SUB	D
2D	DCR	L	60	MOV	H,B	93	SUB	E
2E	MVI	L,byte	61	MOV	H,C	94	SUB	H
2F	CMA		62	MOV	H,D	95	SUB	L
30	SIM[1]		63	MOV	H,E	96	SUB	M
31	LXI	SP,dble	64	MOV	H,H	97	SUB	A
32	STA	adr	65	MOV	H,L	98	SBB	B
33	INX	SP	66	MOV	H,M	99	SBB	C
34	INR	M	67	MOV	H,A	9A	SBB	D
35	DCR	M	68	MOV	L,B	9B	SBB	E
36	MVI	M, byte	69	MOV	L,C	9C	SBB	H
37	STC		6A	MOV	L,D	9D	SBB	L
38	–		6B	MOV	L,E	9E	SBB	M
39	DAD	SP	6C	MOV	L,H	9F	SBB	A
3A	LDA	adr	6D	MOV	L,L	A0	ANA	B
3B	DCX	SP	6E	MOV	L,M	A1	ANA	C
3C	INR	A	6F	MOV	L,A	A2	ANA	D
3D	DCR	A	70	MOV	M,B	A3	ANA	E
3E	MVI	A,byte	71	MOV	M,C	A4	ANA	H
3F	CMC		72	MOV	M,D	A5	ANA	L
40	MOV	B,B	73	MOV	M,E	A6	ANA	M
41	MOV	B,C	74	MOV	M,H	A7	ANA	A
42	MOV	B,D	75	MOV	M,L	A8	XRA	B
43	MOV	B,E	76	HLT		A9	XRA	C
44	MOV	B,H	77	MOV	M,A	AA	XRA	D
45	MOV	B,L	78	MOV	A,B	AB	XRA	E
46	MOV	B,M	79	MOV	A,C	AC	XRA	H
47	MOV	B,A	7A	MOV	A,D	AD	XRA	L
48	MOV	C,B	7B	MOV	A,E	AE	XRA	M
49	MOV	C,C	7C	MOV	A,H	AF	XRA	A
4A	MOV	C,D	7D	MOV	A,L	B0	ORA	B
4B	MOV	C,E	7E	MOV	A,M	B1	ORA	C
4C	MOV	C,H	7F	MOV	A,A	B2	ORA	D
4D	MOV	C,L	80	ADD	B	B3	ORA	E
4E	MOV	C,M	81	ADD	C	B4	ORA	H
4F	MOV	C,A	82	ADD	D	B5	ORA	L
50	MOV	D,B	83	ADD	E	B6	ORA	M
51	MOV	D,C	84	ADD	H	B7	ORA	A
52	MOV	D,D	85	ADD	L	B8	CMP	B
53	MOV	D,E	86	ADD	M	B9	CMP	C
54	MOV	D,H	87	ADD	A	BA	CMP	D
55	MOV	D,L	88	ADC	B	BB	CMP	E
56	MOV	D,M	89	ADC	C	BC	CMP	H
57	MOV	D,A	8A	ADC	D	BD	CMP	L
58	MOV	E,B	8B	ADC	E	BE	CMP	M
59	MOV	E,C	8C	ADC	H	BF	CMP	A
5A	MOV	E,D	8D	ADC	L	C0	RNZ	
5B	MOV	E,E	8E	ADC	M	C1	POP	B
5C	MOV	E,H	8F	ADC	A	C2	JNZ	adr

C3	JMP	adr
C4	CNZ	adr
C5	PUSH	B
C6	ADI	byte
C7	RST	0
C8	RZ	
C9	RET	adr
CA	JZ	adr
CB	–	
CC	CZ	adr
CD	CALL	adr
CE	ACI	byte
CF	RST	1
D0	RNC	
D1	POP	D
D2	JNC	adr
D3	OUT	byte
D4	CNC	adr
D5	PUSH	D
D6	SUI	byte
D7	RST	2
D8	RC	
D9	–	
DA	JC	adr
DB	IN	byte
DC	CC	adr
DD	–	
DE	SBI	byte
DF	RST	3
E0	RPO	
E1	POP	H
E2	JPO	adr
E3	XTHL	
E4	CPO	adr
E5	PUSH	H
E6	ANI	byte
E7	RST	4
E8	RPE	
E9	PCHL	
EA	JPE	adr
EB	XCHG	
EC	CPE	adr
ED	–	
EE	XRI	byte
EF	RST	5
F0	RP	
F1	POP	PSW
F2	JP	adr
F3	DI	
F4	CP	adr
F5	PUSH	PSW
F6	ORI	byte
F7	RST	6
F8	RM	
F9	SPHL	
FA	JM	adr
FB	EI	
FC	CM	adr
FD	–	
FE	CPI	byte
FF	RST	7

[1] Только в 8085.

Для операндов приняты следующие обозначения: adr – адрес; byte – байт; dble – слово (2 байта) – *Прим. перев.*

Таблица А.4. *Команды передачи управления микропроцессоров 8080, 8085*

Условие флага	Переход		Вызов		Возврат	
Знак = истина	JZ	CA	CZ	CC	RZ	C8
Знак = ложь	JNZ	C2	CNZ	C4	RNZ	C0
Перенос = истина	JC	DA	CC	DC	RC	D8
Перенос = ложь	JNC	D2	CNC	D4	RNC	D0
Знак = положительный	JP	F2	CP	F4	RP	F0
Знак = отрицательный	JM	FA	CM	FC	RM	F8
Четность = четная	JPE	EA	CPE	EC	RPE	E8
Четность = нечетная	JPO	E2	CPO	E4	RPO	E0
Безусловное	JMP	C3	CALL	CD	RET	C9

Таблица А.5. *Операции микропроцессоров 8080, 8085 с аккумулятором*

Операция	Код	Функция
XRA A	AF	Очистить A и очистить флаг переноса
ORA A	B7	Очистить флаг переноса
CMC	3F	Инвертировать флаг переноса
CMA	2F	Инвертировать аккумулятор

Операция	Код	Функция
STC	37	Установить флаг переноса
RLC	07	Циклически сдвигать влево
RRC	0F	Циклически сдвигать вправо
RAL	17	Циклически сдвигать влево через флаг переноса
RAR	1F	Циклически сдвигать вправо через флаг переноса
DAA	27	Корректировать аккумулятор в десятичный вид

Таблица А.6. *Операции микропроцессоров 8080, 8085 с парами регистров и стеком*

Команда	PSW (A/F)	Пара регистров			SP	PC	Функция
		B (B/C)	D (D/E)	H (H/L)			
INX		03	13	23	33		Увеличить пару регистров
DCX		0B	1B	2B	3B		Уменьшить пару регистров
LDAX		0A	1A	7E[1]			Загрузить регистр А косвенно (пара регистров содержит адрес)
STAX		02	12	77[2]			Запомнить регистр А косвенно (пара регистров содержит адрес)
LHLD				2A			Загрузить пару регистров H, L прямо (байты 2 и 3 содержат адрес)
SHLD				22			Запомнить пару регистров H, L прямо (байты 2 и 3 содержат адрес)
LXI		01	11	21	31	C3[3]	Загрузить пару регистров непосредственно (байты 2 и 3 содержат непосредственные данные)
PCHL						E9	Загрузить в PC пару регистров H, L (перейти по адресу в H, L)
XCHG			EB				Обменять пары регистров D, E и H, L
DAD		09	19	29	39		Прибавить пару регистров к H, L
PUSH	F5	C5	D5	E5			Записать пару регистров в стек
POP	F1	C1	D1	E1			Получить пару регистров из стека
XTHL				E3			Обменять H, L с вершиной стека
SPHL					F9		Загрузить H, L в SP

[1] Это команда MOV A,M.
[2] Это команда MOV M,A.
[3] Это команда JMP.

Таблица А.7. *Команды рестарта и вводов микропроцессоров 8080, 8085*

Имя команды	Код	Адрес рестарта
RST 0	C7	0000_{16}
RST 1	CF	0008_{16}
RST 2	D7	0010_{16}
RST 3	DF	0018_{16}
RST 4	E7	0020_{16}
TRAP	Аппаратная функция[1]	0024_{16}
RST 5	EF	0028_{16}
RST 5.5	Аппаратная функция[1]	$002C_{16}$
RST 6	F7	0030_{16}
RST 6.5	Аппаратная функция[1]	0034_{16}
RST 7	FF	0038_{16}
RST 7.5	Аппаратная функция[1]	$003C_{16}$

[1] Наличие аппаратной функции является особенностью только микропроцессора 8085.

Таблица А.8. *Справка по ассемблеру микропроцессоров 8080, 8085*

Операторы	Псевдокоманды: общие	Псевдокоманды: перемещаемые	
(,)	ORG	ASEG	NAMA
NUL	END	DSEG	STKLN
LOW, HIGH	EQU	CSEG	STACK
*, /, MOD, SHL, SHR	SET	PUBLIC	MEMORY
+, –	DS	EXTRN	
NOT	DB		
AND	DW		
OR, XOR			

Определение констант	макрокоманды	условное ассемблирование
0BDH, 1AH — шестнадцатеричные	MACRO	IF
105D, 105 — десятичные	ENDM	ELSE
72O, 72Q — восьмеричные	LOCAL	ENDIF
11011B, 00110B — двоичные	REPT	
'TEST', 'A' 'R' — ASCII	IRP	
	IRPC	
	EXITM	

СПРАВКА ПО ПРОГРАММИРОВАНИЮ ДЛЯ ПЕРИФЕРИЙНОГО ИНТЕРФЕЙСА 8255

Конфигурация вводов

8255A

PA3	1	40	PA4
PA2	2	39	PA5
PA1	3	38	PA6
PA0	4	37	PA7
$\overline{RD}$	5	36	$\overline{WR}$
$\overline{CS}$	6	35	RESET
GND	7	34	D_0
A1	8	33	D_1
A0	9	32	D_2
PC7	10	31	D_3
PC6	11	30	D_4
PC5	12	29	D_5
PC4	13	28	D_6
PC0	14	27	D_7
PC1	15	26	V_{CC}
PC2	16	25	PB7
PC3	17	24	PB6
PB0	18	23	PB5
PB1	19	22	PB4
PB2	20	21	PB3

Наименование вводов

$D_7 - D_0$	Шина данных (двунаправленная)
RESET	Сброс
$\overline{CS}$	Выбор устройства
$\overline{RD}$	Чтение
$\overline{WR}$	Запись
A0, A1	Адрес порта
PA7 – PA0	Порт A (разряды)
PB7 – PB0	Порт B (разряды)
PC7 – PC0	Порт C (разряды)
V_{CC}	+5 В
GND	0 В

Рис. Б.1. Назначение выходов микросхемы 8255

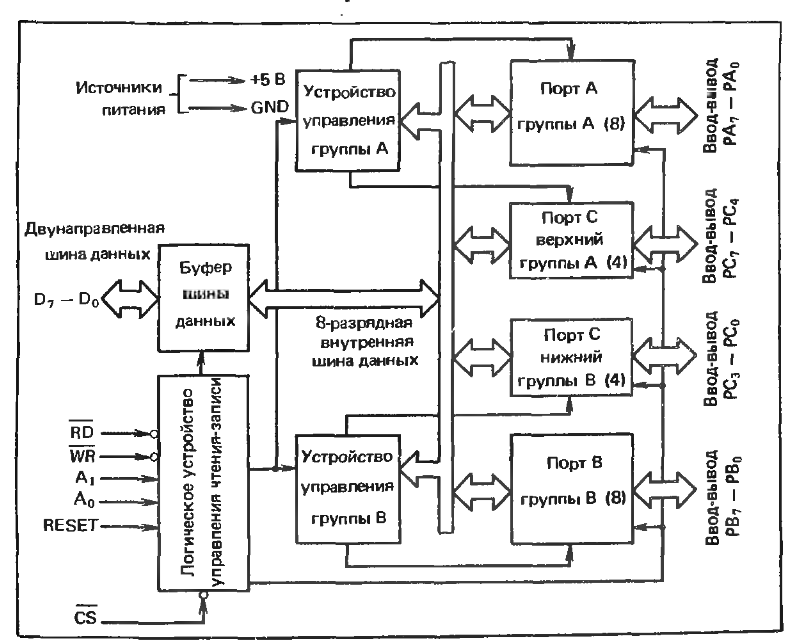


Рис. Б.2. Структурная схема программируемого периферийного параллельного интерфейса 8255

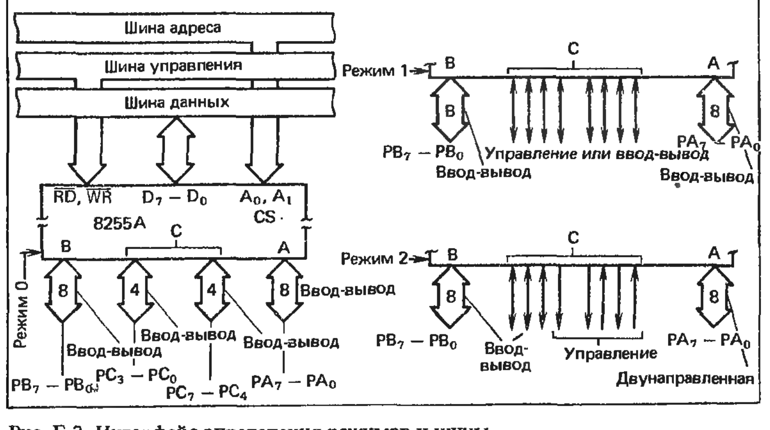


Рис. Б.3. Интерфейс определения режимов и шины

Управляющее слово

D7	D6	D5	D4	D3	D2	D1	D0

Группа B

- D0 → Порт C (нижний): 1 = ввод, 0 = вывод
- D1 → Порт B: 1 = ввод, 0 = вывод
- D2 → Выбор режима: 0 = режим 0, 1 = режим 1

Группа A

- D3 → Порт C (верхний): 1 = ввод, 0 = вывод
- D4 → Порт A: 1 = ввод, 0 = вывод
- D6, D5 → Выбор режима: 00 = режим 0, 01 = ражим 1, 1X = режим 2

D7 → Флаг установки режима: 1 = активный

Рис. Б.4. Формат определения режимов интерфейса 8255

Управляющее слово

D7	D6	D5	D4	D3	D2	D1	D0

x x x

Не используются

Установить — сбросить разряд
1 = установить
0 = сбросить

Выбрать разряд

0	1	2	3	4	5	6	7	
0	1	0	1	0	1	0	1	B_0
0	0	1	1	0	0	1	1	B_1
0	0	0	0	1	1	1	1	B_2

Флаг установки — сброса разряда
0 = активный

Рис. Б.5. Формат установки и сброса разрядов интерфейса 8255

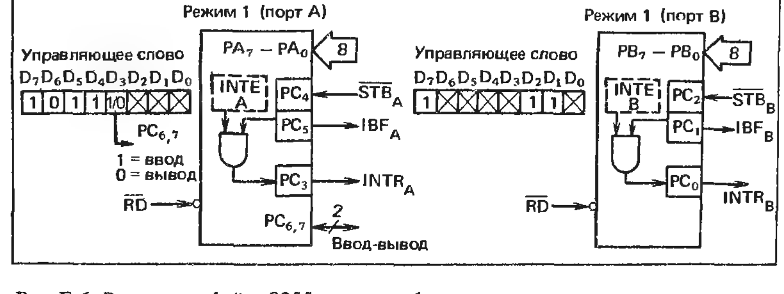


Рис. Б.6. Ввод интерфейса 8255 в режиме 1

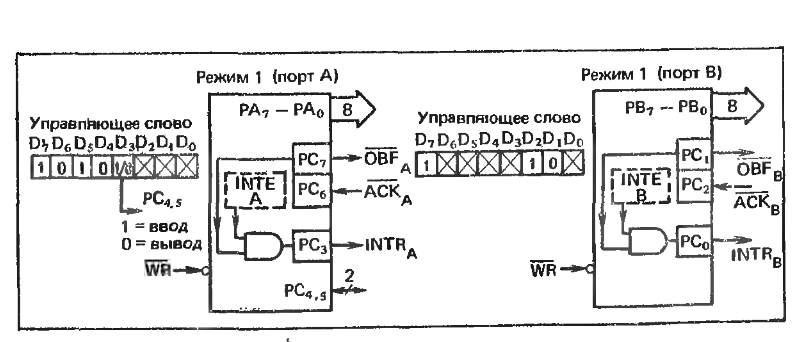


Рис. Б.7. Вывод интерфейса 8255 в режиме 1

Рис. Б.8. Управляющее слово интерфейса 8255 в режиме 2

Рис. Б.9. Двунаправленный режим (режим 2) интерфейса 8255

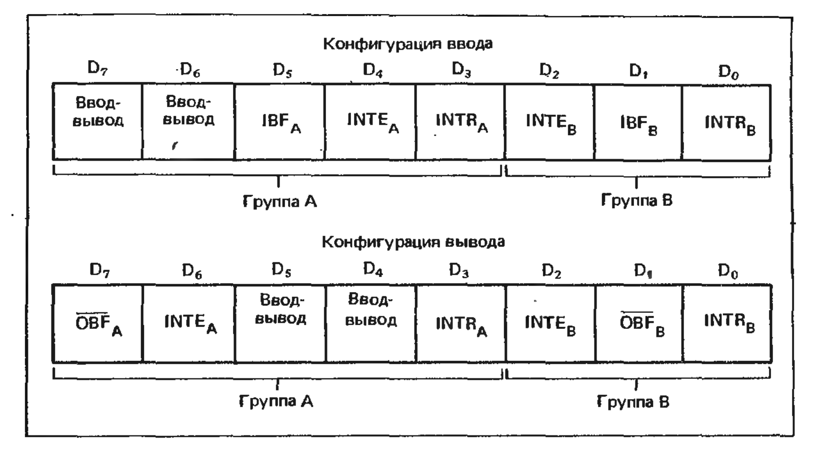


Рис. Б.10. Формат слова состояния интерфейса 8255 в режиме 1

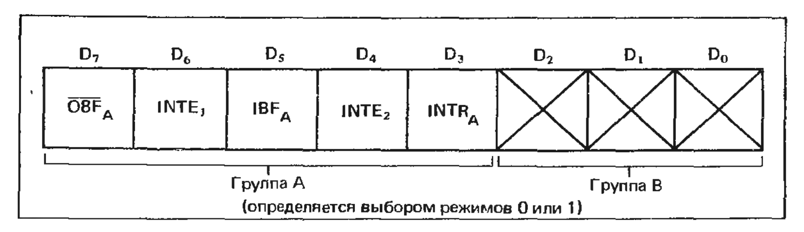


Рис. Б.11. Формат слова состояния интерфейса 8255 в режиме 2

Таблица Б.1. *Операции программируемого периферийного интерфейса 8255*

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Функция
			Операция ввода (чтения)		
0	0	0	1	0	Порт А = шина данных
0	1	0	1	0	Порт В = шина данных
1	0	0	1	0	Порт С = шина данных
			Операция вывода (записи)		
0	0	1	0	0	Шина данных = порт А
0	1	1	0	0	Шина данных = порт В
1	0	1	0	0	Шина данных = порт С
1	1	1	0	0	Шина данных = управление
			Функции, приводящие устройство в нерабочее состояние		
X	X	X	X	1	Шина данных = 3 состояния
1	1	0	1	0	Невыполняемое условие
X	X	1	1	0	Шина данных = 3 состояния

Таблица Б.2. *Назначение портов для периферийного интерфейса 8255 при режиме работы 0*

A		B		Группа А		Номер	Группа В	
D_4	D_3	D_1	D_0	Порт A	Порт C (верхний)		Порт B	Порт C (нижний)
0	0	0	0	Вывод	Вывод	0	Вывод	Вывод
0	0	0	1	Вывод	Вывод	1	Вывод	Ввод
0	0	1	0	Вывод	Вывод	2	Ввод	Вывод
0	0	1	1	Вывод	Вывод	3	Ввод	Ввод
0	1	0	0	Вывод	Ввод	4	Вывод	Вывод
0	1	0	1	Вывод	Ввод	5	Вывод	Ввод
0	1	1	0	Вывод	Ввод	6	Ввод	Вывод
0	1	1	1	Вывод	Ввод	7	Ввод	Ввод
1	0	0	0	Ввод	Вывод	8	Вывод	Вывод
1	0	0	1	Ввод	Вывод	9	Вывод	Ввод
1	0	1	0	Ввод	Вывод	10	Ввод	Вывод
1	0	1	1	Ввод	Вывод	11	Ввод	Ввод
1	1	0	0	Ввод	Ввод	12	Вывод	Вывод
1	1	0	1	Ввод	Ввод	13	Вывод	Ввод
1	1	1	0	Ввод	Ввод	14	Ввод	Вывод
1	1	1	1	Ввод	Ввод	15	Ввод	Ввод

Таблица Б.3. *Краткая сводка режимов работы периферийного интерфейса 8255*

Обозначение вывода микросхемы	Режим работы				
	0		1		2 (только для группы А)
	Ввод	Вывод	Ввод	Вывод	
PA_0	Ввод	Вывод	Ввод	Вывод	↔
PA_1	Ввод	Вывод	Ввод	Вывод	↔
PA_2	Ввод	Вывод	Ввод	Вывод	↔
PA_3	Ввод	Вывод	Ввод	Вывод	↔
PA_4	Ввод	Вывод	Ввод	Вывод	↔
PA_5	Ввод	Вывод	Ввод	Вывод	↔
PA_6	Ввод	Вывод	Ввод	Вывод	↔
PA_7	Ввод	Вывод	Ввод	Вывод	↔
PB_0	Ввод	Вывод	Ввод	Вывод	Не используется
PB_1	Ввод	Вывод	Ввод	Вывод	
PB_2	Ввод	Вывод	Ввод	Вывод	
PB_3	Ввод	Вывод	Ввод	Вывод	
PB_4	Ввод	Вывод	Ввод	Вывод	
PB_5	Ввод	Вывод	Ввод	Вывод	
PB_6	Ввод	Вывод	Ввод	Вывод	
PB_7	Ввод	Вывод	Ввод	Вывод	

Обозначение вывода микросхемы	Режим работы				
	0		1		2 (только для группы А)
	Ввод	Вывод	Ввод	Вывод	
PC_0	Ввод	Вывод	$INTR_B$	$INTR_B$	Ввод-вывод
PC_1	Ввод	Вывод	IBF_B	$\overline{OBF}_B$	Ввод-вывод
PC_2	Ввод	Вывод	$\overline{STB}_B$	$\overline{ACK}_B$	Ввод-вывод
PC_3	Ввод	Вывод	$INTR_A$	$INTR_A$	$INTR_A$
PC_4	Ввод	Вывод	$\overline{STB}_A$	Ввод-вывод	$\overline{STB}_A$
PC_5	Ввод	Вывод	IBF_A	Ввод-вывод	IBF_A
PC_6	Ввод	Вывод	Ввод-вывод	$\overline{ACK}_A$	$\overline{ACK}_A$
PC_7	Ввод	Вывод	Ввод-вывод	$\overline{OBF}_A$	$\overline{OBF}_A$

ПРИЛОЖЕНИЕ В

НАБОР СИМВОЛОВ ASCII

Младшая цифра кода символа		Старшая цифра кода символа							
		0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0	0000	NUL	DLE	SP	0	@	P		p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	”	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	/	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	}
C	1100	FF	FS	,	<	L	\	l	¦
D	1101	CR	GS	–	=	M	]	m	{
E	1110	SO	RS	.	>	N	Λ	n	~
F	1111	SI	US	/	?	O	–	o	DEL

СЛОВАРЬ ТЕРМИНОВ

А

Автоувеличение. Автоматическое увеличение содержимого регистра адреса как часть процесса выполнения команды, использующей этот регистр.

Автоуменьшение. Автоматическое уменьшение содержимого регистра адреса как часть процесса выполнения команды, использующей этот регистр.

Адрес. Идентификационный код, которым отличается одна ячейка памяти или порт ввода-вывода от других и который может быть использован для выбора определенной ячейки памяти или порта ввода-вывода.

Адрес абсолютный. Адрес, который определяет ячейку памяти или устройство ввода-вывода без использования базы, смещения или другого фактора. *См. также* Адрес эффективный, Смешение относительное.

Адрес базовый. Адрес памяти, с которого начинается массив или таблица. Называется также начальным адресом или базой.

Адрес страничный. Идентификатор, характеризующий конкретный адрес памяти на известной странице. В ЭВМ, ориентированных на работу с байтами, это обычно младшие восемь разрядов адреса памяти.

Адрес устройства. Адрес порта, связанного с устройством ввода-вывода.

Адрес эффективный. Действительный адрес, используемый в команде при выборке и запоминании данных.

Адресация абсолютная. Способ адресации, при котором команда содержит действительный адрес, необходимый для ее выполнения; является обратной способам адресации, при которых команда содержит относительное смещение или определяет базовый адрес.

Адресация индексная. Способ адресации, при котором эффективный адрес получается модификацией адреса с помощью индексного регистра.

Адресация косвенная. Способ адресации, при котором эффективный адрес находится по адресу, являющемуся частью команды.

Адресация косвенная индексная. См. Послеиндексирование.

Адресация непосредственная. Способ адресации, при котором данные, необходимые для выполнения команды, являются частью команды. Эти данные следуют в памяти непосредственно за кодом операции.

Адресация относительная. Способ адресации, при котором адрес, задаваемый в команде, определяет смешение относительно базового адреса.

Адресация программы относительная. Форма относительной адресации, при которой базовым адресом является значение счетчика команд. Использование этой формы адресации облегчает перемещение программы из одной области памяти в другую.

Адресация прямая. Способ адресации, при котором команда содержит адрес, необходимый для ее выполнения.

Адресация регистровая косвенная. Способ адресации, при котором адрес, необходимый для выполнения команды, содержится в регистре.

Адресное пространство. Общий диапазон адресов, к которым может обращаться ЭВМ.

Аккумулятор. Регистр, являющийся источником операнда и местом назначения для результата в большинстве арифметических и логических операций.

Активный переход. Фронт импульса, устанавливающего индикатор. Возможны отрицательный (переход из 1 в 0) или положительный (переход из 0 в 1) фронты.

Антипереполнение (стека). Результат чтения из стека большего числа данных, чем было в него записано.

Арифметическо-логическое устройство. Устройство, которое может выполнять различные арифметические и логические функции; по входным данным выбирается конкретная функция, выполняемая устройством во время определенного такта.

Асинхронная работа. Работа, при которой не происходит обращения к внешнему источнику, задающему время, т. е. работа с неравномерными интервалами.

Ассемблер. Программа для ЭВМ, преобразующая программу на языке ассемблера в форму (на машинном языке), в которой ЭВМ может прямо ее выполнять. Ассемблер транслирует мнемонические коды операций и имена в их числовые эквиваленты и присваивает данным и командам ячейки в памяти. Для операционной системы CP/M распространенным является ассемблер MAC фирмы Digital Research.

Б

Байт. Элемент из восьми разрядов. Иногда говорят, что байт содержит старшую половину или цифру (старшие по значению четыре разряда байта) и младшую половину или цифру (младшие по значению четыре разряда).

Блок. Полная группа или раздел, например набор регистров или раздел памяти.

Блок управления вводом-выводом (IOCB). Группа ячеек памяти, которая содержит информацию, необходимую для управления устройствами ввода-вывода. Обычно в эту информацию включаются адреса подпрограмм, выполняющих такие операции, как передача одного элемента данных или определение состояния устройства.

Бод. Мера скорости передачи последовательных данных; число разрядов в секунду, включая разряды данных и разряды, используемые для синхронизации, проверки ошибок и других целей. Распространены следующие скорости в бодах: 110, 300, 600, 1200, 2400, 4800, 9600 и 19200.

Булева переменная. Переменная, имеющая только два возможных значения, которые могут быть представлены как истина и ложь, или 1 и 0. *См. также* Флаг.

Буфер. Временная область хранения данных перед их передачей в конечное место назначения.

Буфер заполнен. Сигнал, активный в случае, когда буфер полностью занят данными, которые еще не переданы в их конечное место назначения.

Буфер пустой. Сигнал, активный в случае, когда все данные, записанные в буфер, уже переданы в их конечное место назначения.

В

Ввод-вывод изолированный. Способ адресации портов ввода-вывода, при котором используется система декодирования, не входящая в систему декодирования памяти. Таким образом, порты ввода-вывода не занимают адреса памяти.

Ввод-вывод, отраженный на память. Способ адресации портов ввода-вывода, при котором используется тот же метод декодирования, что и в разделе памяти. При этом порты ввода-вывода занимают адреса памяти.

Ввод-вывод программный. Ввод и вывод, выполняемые под управлением программы без использования прерываний или каких-либо других специальных аппаратных методов.

Вектор прерываний. Адрес, по которому передается управление ЭВМ при прерывании; обычно это начальный адрес обслуживающей программы.

Вершина стека. Адрес, содержащий элемент, записанный в стек последним.

Вложение. Иерархическая организация программы, при которой один уровень содержится внутри другого и т. д. Уровень вложения — это число передач управления, необходимое для того, чтобы достичь определенной части программы без возврата на более высокий уровень.

Возврат (из подпрограммы). Передача управления в программу, вызвавшую подпрограмму, и возобновление ее выполнения.

Временная область. Область памяти, которую обычно легко использовать для запоминания переменных данных и промежуточных результатов.

Время выполнения команды. Время, необходимое для выборки, декодирования и выполнения команды.

Выборка команды. Процесс адресации памяти и чтения команды центральным процессором для декодирования и выполнения.

Вывод данных. Выдача полного раздела памяти или группы регистров на устройство вывода.

Вызов (подпрограммы). Передача управления подпрограмме, при которой сохраняется информация, необходимая для возобновления выполнения текущей программы. Вызов отличается от передачи управления тем, что при вызове запоминается предыдущее положение в программе, а при переходе — нет.

Г

Генератор скорости передачи. Генерирует соответствующие интервалы времени между разрядами для последовательной передачи данных.

Глобальная переменная. Переменная, которая определена больше чем в одном разделе программы.

Готовность для данных. Сигнал, указывающий на готовность приемника получать следующие данные.

Граница слова. Граница между 16-разрядными элементами, содержащими два байта информации. Если информация запоминается элементами длиной в слово, то она содержится только в парах байтов, выравненных по границам слов. Пары байтов, невыравненные подобным образом, содержат один байт из одного слова и один байт из другого.

Д

Данные готовы. Сигнал, который, будучи активным, указывает, что для получателя готовы новые данные.

Данные приняты. Сигнал, который устанавливается, когда приняты самые последние данные.

Длина команды. Размер памяти, необходимый для запоминаиия всей команды.

Длина в байт. Длина элемента, равная восьми разрядам.

Длина в слово. Длина элемента, равная 16 разрядам.

Дополнение. Инвертирование. *См. также* Дополнение до одного, Дополнение до двух.

Дополнение до двух. Двоичное число, которое при сложении в двоичном сумматоре с исходным числом дает нуль. Дополнение до двух может быть получено вычитанием числа из нуля или добавлением 1 к дополнению до одного.

Дополнение до девяти. Результат вычитания десятичного числа из числа, состоящего только из девяток.

Дополнение до десяти. Результат вычитания десятичного числа из нуля (игнорируется отрицательный знак) или добавления 1 к дополнению до девяти.

Дополнение до одного. Логическое дополнение числа, получаемого в результате замены каждого 0 на 1 и каждой 1 на 0.

Драйвер ввода-вывода. Программа передачи данных на устройство ввода-вывода или получения от него данных. Называют также *драйвером* или *вспомогательной программой ввода-вывода.* Наряду с физической передачей данных драйвер должен также выполнять функции инициализации, управления и получения информации о состоянии устройства.

З

Заголовок очереди. Набор ячеек памяти, содержащих описание текущего положения и состояния очереди.

Задача. Программа, которая может выполняться под управлением супервизора как часть системы.

Заем. Разряд, который устанавливается в 1, если результат вычитания отрицательный, и в 0, если результат положительный или равен 0. Заем используется обычно при вычитании чисел, которые слишком велики для того, чтобы их можно было обработать в одной операции.

Заем инвертированный. Разряд, который установлен в 0, когда результат вычитания отрицательный, и в 1, если результат положительный или равен 0. Инвертированный заем может использоваться так же, как обычный заем, за тем исключением, что для расширения числа необходимо дополнение инвертированного заема (т. е. 1 минус его значение).

Закрыть (файл). Сделать файл неактивным. В результате файл содержит последнюю информацию, записанную в него пользователем. Обычно после работы с файлом пользователь должен закрывать его.

Записать в стек. Запомнить операнд в стеке.

Заполнить. Поместить данные в область памяти, которая до этого не использовалась, инициализировать память.

Запретить. Остановить какую-либо операцию или распознавание сигналов (например, прерывания).

Запрос прерывания. Сигнал, который является активным, когда периферийное устройство запрашивает обслуживание; часто используется для того, чтобы вызвать прерывание центрального процессора.

И

Импульс сигнала времени. Регулярный сигнал времени, управляющий переходами в системе.

Индекс. Элемент данных, используемый для идентификации определенного элемента массива или таблицы.

Индекс буфера. Индекс следующего доступного адреса в буфере.

Интерполяция. Вычисление значения функции между точками, значения в которых уже известны.

Интерфейс параллельный. Интерфейс между центральным процессором и устройствами ввода или вывода, осуществляющий обработку данных параллельно (передается больше одного разряда за один раз). В семействе 8080, 8085 им является программируемый параллельный интерфейс 8255.

Интерфейс последовательный. Интерфейс между центральным процессором и устройством ввода или вывода, который передает данные последовательно. В семействе 8080, 8085 широко применяется последовательный программируемый интерфейс связи 8251.

К

Клавиша функциональная. Клавиша, при нажатии на которую система выполняет некоторую функцию (такую как очистка экрана на видеотерминале) или процедуру.

Код двоично-десятичный. Представление десятичных чисел, при котором каждая десятичная цифра кодируется отдельно двоичным числом.

Кодирование. Написание команд на языке ЭВМ.

Код исправления ошибок. Код, который получатель может использовать для исправления ошибок в сообщениях; сам по себе этот код не содержит дополнительных сообщений.

Код исходный (или *исходная программа*). Программа, написанная на языке ассемблера или на языке высокого уровня.

Код объектный (или *объектная программа*). Программа, которая является результатом работы транслятора, например ассемблера; обычно это программа на машинном языке, готовая к выполнению.

Код операции. Часть команды, определяющая операцию, которую необходимо выполнить.

Код условия. См. Флаг.

Код BCD стандартный (или *8, 4, 2, 1*). Представление в коде BCD, при котором позиции разрядов имеют те же самые веса, что и в обычных двоичных числах.

Команда. Группа разрядов, определяющая операцию ЭВМ; является частью набора команд.

Команда перехода. Команда, по которой в счетчик команд помещается новое значение вместо нормального увеличения его на единицу за шаг. Команды перехода могут быть условными, т. е. новое значение может помещаться в счетчик команд только в том случае, если будет удовлетворено некоторое условие.

Команда сдвига. Команда пересылки всех разрядов данных на определенное число разрядов, как в регистре сдвига.

Команда трансляции. Команда преобразования своего операнда в соответствующую запись в таблице.

Комментарий. Раздел программы, который не выполняет никаких других функций, кроме документирования. При трансляции комментарии не обрабатываются, а просто копируются в листинг программы.

Конец очереди. Положение самого старого элемента в очереди, т. е. элемента, записанного раньше других.

Код определения ошибки. Код, который может служить для определения ошибок в сообщениях; сам код не содержит никакой дополнительной информации.

Контроллер протокола. Микросхема, выполняющая все или многие функции, необходимые для протокола. Программируемый контроллер протокола 8273 служит в качестве такого контроллера в семействе 8080, 8085.

Л

Линеаризация. Математическая аппроксимация функции с помощью прямой линии, проведенной между двумя точками, значения в которых известны.

Логика мажоритарная. Функция комбинаторной логики, которая истинна, когда истинны более половины входных величин.

Логика отрицательная. Обеспечивается элементами, в которых состояние логического нуля является активным.

М

Маркер. Состояние 1 на линии связи последовательных данных.

Маска. Набор двоичных разрядов, который служит для выделения одного разряда или нескольких из группы разрядов.

Маска прерываний. Разряд, определяющий, какие из прерываний могут быть распознаны. Чтобы разрешить прерывания, маска (или разряд) запрета должна быть очищена, в то время как разряд разрешения должен быть установлен.

Массив. Набор связанных элементов данных, запоминаемый обычно в последовательных адресах памяти.

Математическое обеспечение системное. Программы, предназначенные для выполнения административных функций или для целей отладки других программ.

Метка. Имя, придаваемое команде или оператору программы, идентифипирующее положение в памяти кодов машинного языка, или значение, присвоенное данной командой или оператором.

Микропроцессор. Функционально законченный центральный процессор ЭВМ, сконструированный из одной ипи нескольких интегральных микросхем.

Микро-ЭВМ. ЭВМ, центральным процессором которой служит микропроцессор.

Мнемоника. Имя, которое подсказывает действительное значение или назначения того объекта, к которому она относится.

Многозадачность. Выполнение многих задач в течение одного периода времени с помощью приостановки выполнения задач, ожидающих ввода, вывода, завершения других задач или внешнего события.

Модуль. Часть или раздел программы.

Монитор. Программа, позволяющая пользователю ЭВМ вводить программы и данные, запускать программы в работу, проверять содержимое памяти ЭВМ и регистров и использовать периферийные устройства ЭВМ. *См. также* Операционная система.

Н

Начало (очереди). Положение элемента, введенного в очередь последним.

Нет операции. Команда, согласно которой не делается ничего, кроме увеличения счетчика команд.

Номер страницы. Идентификатор, характеризующий конкретную страницу в памяти. В ЭВМ, в которых адресуемой единипей является байт, это обычно восемь старших по значению разрядов адреса памяти.

Нормализация (чисел). Приведение числа к нормальному или стандартному формату. Типичным примером явпяется масштабирование двоичной дроби к значению, при котором старший по значению байт равен 1.

О

Обновление. Перезапись данных в память перед тем, как ее содержимое теряется. Динамические ОЗУ должны периодически обновпяться (обычно каждые несколько миллисекунд), иначе их содержимое может быть самопроизвольно утеряно.

Объединение. Помещение вместе. При работе со строками под объединением понимают помещение одной строки после другой.

Объем памяти. Общее число адресов памяти (определяемое обычно в байтах), которое может быть использовано в конкретной ЭВМ.

Организация буфера циклическая. Такая организация, при которой концы буфера соединены между собой.

Оперативное запоминающее устройство (ОЗУ). Память, из которой при нормальной работе можно считывать информацию и которую можно изменять (записывать информацию).

Операционная система (ОС). Программа управления всеми операциями ЭВМ и выполнения таких функпий, как отведение места в памяти для программ и данных, планирование порядка выпопнения программ, обработка прерываний и управление всей системой ввода-вывода. Известна также как монитор, диспетчер или главная управляющая программа, хотя термин монитор обычно применяется по отношению к простым операционным системам с ограниченными функциями.

Операционная система реального времени. Операционная система, служащая супервизором для программ, выполняемых в реальном времени. Называют также диспетчером реального времени или монитором реального времени.

Открыть (файл). Привести файл в состояние готовности для последующего использования. Перед работой с файлом пользователь обычно должен открыть его.

Отладка. Нахождение и исправление ошибок в программе.

Отладчик. Системная программа, которая позволяет пользователю находить и исправлять ошибки в его программах. Некоторые версии отладчика называют инструментом динамической отладки, или DDT (dynamic debugging tool). Для ОС CP/M широко используется символьный отладчик команд или SID (Symbolic Instruction Debugger) фирмы Digital Research.

Опрос. Определение готовности устройств с помощью последовательной проверки их состояний.

Очередь. Набор задач, адресов памяти или других элементов, которые используются в порядке "первым пришел – первым вышел", т. е. первый элемент, поступивший в очередь, используется или удаляется первым.

Очистить. Установить в нуль.

Очистка стека. Удаление ненужных элементов из стека; обычно выполняется с помощью изменения указателя стека.

П

Память энергозависимая. Память, содержимое которой при выключении питания теряется.

Память энергонезависимая. Память, содержимое которой при отключении питания сохраняется.

Параметр. Элемент, предоставляемый для обработки программе или подпрограмме.

Пара регистров. Для 8080, 8085 – два 8-разрядных регистра, к которым можно обращаться как к 16-разрядному элементу.

Пауза. Нулевое состояние в линии последовательной передачи данных.

Передача параметров. Предоставление параметров подпрограмме.

Перенос. Разряд, равный 1, если при сложении происходит переполнение, значение которого переносится в следующую цифру.

Поиск двоичный. Способ поиска, при каждой итерации которого набор элементов делится на две равные (или приблизительно равные) части. Затем находится часть, содержащая искомый элемент, и используется в качестве набора элементов для следующей итерации. Таким образом, при каждой итерации размер набора элементов, в котором производится поиск, уменьшается вдвое. Этот метод применим для упорядоченного набора элементов.

Поле. Набор из одной или более позиций внутри более крупного элемента, такого как байт, слово или запись.

Получение из стека. Удаление операнда из стека.

Порт. Основная адресуемая единица в системе ввода-вывода ЭВМ.

Порядок по строкам. Способ хранения элементов многомерных массивов в памяти, при котором индексы изменяются, начиная с самого правого. Это значит, что если типичный элемент выглядит как A (I, J, K), а элементы начинаются с A (0, 0, 0), то порядок будет следующим: A (0, 0, 0), A (0, 0, 1), . . . , A (0, 1, 0), A (0, 1, 1), . . . Противоположный способ (при котором первым изменяется самый левый индекс) называется порядком по столбцам.

Послеиндексирование. Способ адресации, при котором для определения эффективного адреса сначала получается косвенно базовый адрес, а затем производится индексирование относительно этого базового адреса. Приставка "после" указывает на тот факт, что индексирование выполняется после косвенного получения адреса.

Перенос вспомогательный. Флаг, используемый в 8-разрядных ЭВМ для указания на перенос из младшей (4-разрядной) цифры.

Переполнение (стека). Превышение объема памяти, отведенной под стек.

Переполнение при дополнении до двух. Ситуация, при которой результат арифметической операции над числами со знаком не может быть правильно представлен, т. е. значащая часть числа переполняется, распространяясь в знаковый разряд.

Пересылка блока. Пересылка целого набора данных из одной области памяти в другую.

Переход косвенный. Команда, передающая управление по адресу, который хранится в регистре или ячейки памяти.

Периферийное устройство готово. Сигнал, который активен, когда периферийное устройство готово принять еще данные.

Планировщик. Программа определения начала и окончания работы другой программы.

Поддержка (программы). Изменение и исправление программ, которые находятся в эксплуатации.

Подпрограмма. Программа, которая может быть вызвана на исполнение более чем из одного места главной программы.

Подтверждение. Асинхронная передача, при которой передающая и получающая стороны обмениваются сигналами, чтобы синхронизировать обмен и установить состояние передачи данных. Обычно передатчик указывает, что новые данные доступны, а приемник считывает эти данные и указывает на готовность читать следующие.

Подтверждение готовности данных. Сигнал, который активен, когда новые данные доступны для получателя.

Поиск блока. См. Сравнение блока.

Послеувеличение. Увеличение содержимого регистра адреса после его использования.

Постоянное запоминающее устройство (ПЗУ). Память, из которой при нормальных операциях можно только прочитать данные, но нельзя изменить.

Предындексация. Способ адресации, при котором для определения эффективного адреса сначала производится индексирование относительно базового адреса, а затем косвенно используется индексированный адрес. Приставка "пред" указывает на тот факт, что индексирование выполняется до косвенного получения адреса. Само собой разумеется, массив, начинающийся с заданного базового адреса, должен содержать адреса, которые могут быть использованы косвенно.

Предувеличение. Увеличение содержимого регистра адреса до его использования.

Предуменьшение. Уменьшение содержимого регистра адреса до его использования.

Прерывание. Временная приостановка нормальной последовательности операций ЭВМ и передача управления специальной программе.

Прерывание маскируемое. Прерывание, которое может быть запрещено системой.

Прерывание немаскируемое. Прерывание в центральном процессоре, которое не может быть запрещено.

Прерывание по отказу питания. Прерывание, которое информирует центральный процессор о предстоящем отключении источника питания.

Прерывание с использованием вектора. Прерывание, при котором вырабатывается идентификационный код (или вектор), используемый центральным процессором для передачи управления соответствующей обслуживающей программе.

Приостановить (задачу). Остановить выполнение и сохранить состояние задачи до некоторого будущего времени.

Проверка по избыточности циклическая (CRC). Генерируется код определения ошибок с использованием полинома, этот код может быть добавлен к данным.

Проверка разряда. Операция проверки разряда на равенство 0 или 1. Обычно используются логической операцией И с соответствующей маской.

Программа библиотечная. Программа, являющаяся частью набора программ, записанная и документированная в соответствии со стандартным форматом.

Программа вспомогательная. Программа общего назначения, поставляемая обычно вместе с ЭВМ как часть операционной системы; обычно служит для выполнения стандартных или общих операций, таких как сортировка, преобразование данных из одного формата в другой или копирование файла.

Программа диагностическая. Программа, предназначенная для проверки устройства и выдачи сообщения о его работе.

Программа задержки. Программа, единственная функция которой состоит в том, чтобы расходовать машинное время.

Программа обслуживания прерываний. Программа, при выполнении которой производятся действия, необходимые в ответ на прерывание.

Программа прозрачная. Программа, выполняемая без взаимодействия с операциями других программ.

Программирование модульное. Метод программирования, при котором программа делится на логически самостоятельные разделы, или модули.

Программируемая периферийная интегральная микросхема (или *программируемый периферийный интерфейс*). Интегральная микросхема, которая может работать в различных режимах; ее текущий режим работы определяется с помощью программной загрузки регистра управления. Для семейства 8080, 8085 — это программируемый периферийный параллельный интерфейс 8255.

Программируемый интерфейс связи (PCI). В семействе 8080, 8085 — это последовательный интерфейс 8251.

Программируемый таймер. Устройство, которое под управлением программы может выполнять различные задачи, связанные с временем, включая генерацию задержек. В семействе 8080, 8085 программируемым таймером является программируемый интервальный таймер 8253.

Протокол. Соглашения, определяющие формат и временные характеристики обмена данными между системами связи.

Псевдооперация (или *псевдокоманда*). Операция на языке ассемблера, вызывающая действия, в результате которых не генерируются команды на машинном языке.

Процессор центральный. Управляющая часть ЭВМ, контролирующая работу машины, загрузку и выполнение команд и выполнение арифметических и логических функций.

Р

Работа по прерываниям. Работа, зависящая от прерываний; пока не будет получен сигнал прерывания программа может находиться в сосотоянии бездействия.

Размер (размерности массива). Расстояние в памяти между элементами данной размерности, следующими друг за другом; число байтов между начальным адресом некоторого элемента и начальным адресом элемента, индекс которого в данной размерности на единицу больше при тех же самых индексах в других размерностях.

Разрешение. Разрешение какой-либо деятельности или рапознавания сигнала (например, прерывания).

Разряд, младший по значению. Самый правый разряд группы разрядов, т. е. разряд 0 байта или 16-разрядного слова.

Разряд стартовый. Одиоразрядный сигнал, указывающий на начало передачи данных в асинхронном устройстве.

Разряд, старший по значению. Самый левый разряд в группе разрядов, т. е. разряд 7 в байте или разряд 15 в 16-разрядном слове.

Разряд стоповый. Одноразрядный сигнал, указывающий для асинхронного устройства на конец передачи данных.

Разряд четности. Одноразрядный код, указывающий на ошибку, добавляемый для того, чтобы сделать общее число разрядов, включая разряд четности, четным (четная четность) или нечетным (нечетная четность). Называют также ***вертикальной четностью*** или ***вертикальной проверкой по избыточности*** (VRC).

Распределение (памяти) динамическое. Предоставление памяти для подпрограммы в момент ее вызова. Альтернативой является статическое распределение фиксированной области для каждой подпрограммы. Динамическое распределение часто снижает общий объем требуемой памяти благодаря совместному использованию области памяти под-

программами; однако обычно это требует дополнительного времени для выполнения и управления распределением памяти.

Распределение (памяти) статическое. Присвоение фиксированной области памяти для данных и программ; альтернативой является динамическое распределение, при котором область памяти выделяется по мере необходимости.

Расширение знака. Процесс копирования знакового (старшего по значению) разряда, как это делается при арифметическом сдвиге. Расширение знака сохраняет знак при делении или нормализации чисел, являющихся дополнением до двух.

Расширение числа. Добавление цифры к числу без изменения его значения, для того чтобы оно соответствовало некоторому формату. Например, с помощью нулей можно расширить 8-разрядный результат без знака, чтобы заполнить 16-разрядное слово.

Реальное время. Отсчитывается при синхронизации по действительному появлению событий.

Регистр. Ячейка внутри процессора для хранения данных.

Регистр адреса. Регистр, который содержит адрес памяти.

Регистр индексный. Регистр, который может быть использован для модификации адресов памяти.

Регистр командный. См. Регистр управляющий.

Регистр направления данных. Регистр, определяющий, будут ли использованы двунаправленные линии ввода-вывода для ввода или для вывода.

Регистр, разрешающий только запись в него. Регистр, который может изменяться центральным процессором, но содержимое которого не может быть прочитано. Если в программе необходимо знать состояние такого регистра, то она должна хранить копию помещаемых в него данных.

Регистр состояния. Регистр, содержимое которого указывает на текущее состояние или режим работы устройства.

Регистр управляющий. Регистр, содержимое которого задает состояние передачи или способ работы устройства.

Регистр флагов. Регистр, который содержит все флаги. Его называют также *регистром состояния (процессора).*

Редактор. Программа обработки текста, позволяющая пользователю вносить поправки, дополнения, осуществлять удаление и другие изменения. Для CP/M широко распространен редактор ED фирмы Digital Research.

Реентерабельная (программа или подпрограмма). Может выполняться в то же самое время, когда ее выполнение прервано или же она находится по какой-либо причине в состоянии ожидания.

Режим автоматический (для периферийных интегральных микросхем). Режим работы, при котором периферийная интегральная микросхема автоматически выдает управляющие сигналы без специального программного вмешательства.

Режим свободной работы. Режим работы таймера, при котором таймер указывает на окончание интервала времени и затем начинает отсчитывать новый интервал той же самой длительности. Этот режим называется также непрерывным режимом работы.

С

Связь подпрограммы. Механизм, при котором ЭВМ сохраняет информацию, необходимую для возобновления работы текущей программы после окончания выполнения подпрограммы.

Сдвиг арифметический. Операция сдвига, при которой знаковый (старший по значению) разряд остается без изменения. В результате при сдвиге вправо копии знакового разряда пересылаются вправо (это называется расширением знака).

Сдвиг логический. Операция сдвига, при которой в тот конец, из которого сдвигаются исходные данные, помещаются нули.

Сдвиг циклический. Операция сдвига, выполняемая так, будто данные расположены по кольцу, т. е. старший и младший по значению разряды находятся рядом.

Сделать отрицательным. Найти дополнение числа до двух.

Сигнал состояния. Сигнал, указывающий на текущее состояние передачи или режим работы устройства.

Символ синхронизации. Символ, используемый только для синхронизации передатчика и приемника.

Синхронная работа. Работа по единому источнику времени, т. е. в согласованные интервалы времени.

Система команд. Набор команд общего назначения, доступных на данной ЭВМ. Набор вводимых команд, на которые центральный процессор должен известным образом реагировать при их выборке, декодировании и выполнении.

Система опроса прерываний. Система прерываний, в которой программа определяет источник конкретного прерывания, последовательно проверяя состояния потенциальных источников прерывания.

Система прерываний с приоритетами. Система прерываний, в которой некоторые прерывания имеют более высокий приоритет, т. е. они могут быть обслужены первыми или же могут прервать выполнение других обслуживающих программ.

Система управления вводом-выводом (IOCS). Набор программ, предназначенных для управления выполнением операций ввода-вывода.

Слово. Основное поле разрядов, которые ЭВМ может обрабатывать одновременно. Когда речь идет о микропроцессорах, этот термин часто относят к 16-разрядному элементу данных.

Слово двойное. В приложении к микропроцессорам – 32-разрядный элемент.

Слово состояния процессора (PSW). Пара регистров микропроцессора 8080 или 8085, содержащая аккумулятор (старший по значению байт) и флаги (младший по значению байт).

Смещение относительное. Разность между действительным адресом, который должен быть использован в программе, и текущим значением счетчика команд.

Сортировка пузырьковым методом. Метод последовательной сортировки элементов массива, при котором соседние элементы, если они не стоят в нужном порядке, меняются местами.

Состояние задачи. Набор параметров, характеризующих текущее состояние задачи. Выполнение задачи может быть приостановлено и вновь продолжено, поскольку ее состояние сохраняется и восстанавливается.

Список. Упорядоченный набор элементов.

Список связанный. Список, каждый элемент которого содержит указатель на другой элемент. Называют также *цепочкой* или *цепочным списком.*

Способ адресации. Способ задания адресов, который должен быть использован при выполнении команды. Обычными способами адресации являются следующие: прямая, непосредственная, индексная, косвенная и относительная.

Сравнение блока. Просмотр блока памяти до тех пор, пока не будет найден искомый элемент или же не будет проверен весь блок.

Ссылка внешняя. Использование в программе имени, которое определено в другой программе.

Стек. Раздел памяти, доступ к которому возможен только по способу "последним пришел – первым ушел". Это значит, что данные могут добавляться в стек или удаляться из стека только через его вершину; новые данные помещаются над старыми, а удаление элемента данных делает элемент, лежащий ниже, новой вершиной.

Стек аппаратный. Стек, автоматически управляемый ЭВМ при выполнении команд, его использующих.

Стек программный. Стек, работа с которым выполняется с помощью команд в противоположность аппаратному стеку, который автоматически управляется ЭВМ.

Строб. Сигнал, идентифицирующий еще один набор сигналов; может быть использован для управления буфером, фиксатором или регистром.

Строка. Массив (набор данных), состоящий из символов.

Сумма контрольная. Логическая сумма, которая включается в блок данных, чтобы избежать ошибок при записи или передачи.

Сумма логическая. Двоичная сумма без переносов из разряда в разряд. *См. также* Сумма контрольная, функция ИСКЛЮЧАЮЩЕЕ ИЛИ.

Строка. Массив (набор данных), состоящий из символов.

Счетчик команд (регистр PC). Регистр, содержащий адрес команды, которая должна быть следующей выбрана из памяти.

Т

Таблица переходов. Таблица, содержащая начальные адреса выполняемых программ, используемая для передачи управления одной из них.

Таблица справочная. Массив данных, организованный таким образом, что ответ на задачу может быть получен просто с помощью выбора необходимой записи (без каких-либо вычислений).

Таблица устройств ввода-вывода. Таблица, устанавливающая соответствие между логическими номерами устройств, к которым программы обращаются, и физическими устройствами, которые в действительности используются для передачи данных.

Тактовый импульс. Регулярный сигнал, управляющий временем передачи в системе.

Телетайп. Устройство, содержащее клавиатуру и осуществляющее последовательную печать, используемое часто для связи и совместно с ЭВМ.

Телетайп стандартный. Телетайп, работающий асинхронно со скоростью 10 символов в секунду.

Терминатор. Элемент данных, единственной функцией которого является указание на конец массива.

Тест с перемещающимся разрядом. Процедура, при которой единичный разряд, равный 1, перемещается по всем позициям разрядов в памяти; при этом проверяется, правильно ли разряд считывается из памяти.

Точка прерывания. Задаваемое пользователем условие, при котором выполнение программы временно заканчивается; используется для отладки программ. Задание условий, при которых должно закончиться выполнение программы, называют установкой точки прерывания, а удаление этих условий называют очисткой точки прерывания.

Трассировка. Средство отладки, которое выдает информацию о ходе выполнения программы. При трассировке обычно печатаются все или некоторые промежуточные результаты.

У

Указатель. Место в памяти, где хранится адрес элемента данных, а не сам элемент, т. е. указывается, где расположен элемент.

Указатель буфера. Ячейка в памяти, содержащая следующий доступный адрес в буфере.

Указатель стека. Регистр, содержащий адрес вершины стека.

Устройство логическое. Устройство ввода или вывода, к которому обращается программа. Действительное физическое устройство можно определить по таблице устройств ввода-вывода, содержащей необходимые адреса ввода-вывода (или начальные адреса драйверов ввода-вывода), соответствующие логическим номерам устройств.

Устройство многофункциональное. Устройство, выполняющее в вычислительной системе более одной функции; этот термин обычно относится к устройствам, содержащим память, порты ввода-вывода, таймеры и т. д. В семействе 8080, 8085 популярны такие многофункциональные устройства, как ОЗУ + УВВ + счетчик-таймер 8155 или 8156, ПЗУ + УВВ 8355 и стираемое ППЗУ + УВВ 8755.

Устройство ввода-вывода программируемое. Устройство ввода-вывода, режим работы которого задается с помощью программной загрузки регистров.

Устройство физическое. В противоположность логическому устройству это действительное устройство ввода или вывода.

Ф

Файл. Набор связанной информации, которая при записи или получении рассматривается как единое целое.

Флаг (или *код условия,* или *разряд состояния*). Одиночный разряд, указывающий на некоторые условия в ЭВМ; часто используется для выбора между альтернативными последовательностями команд.

Флаг (программный). Индикатор, который может быть включен или выключен; может служить для выбора решения при наличии двух возможных способов действия. Другие термины с тем же значением — булева переменная и семафор.

Флаг знака. Флаг, содержащий старший по значению разряд результата предыдущей операции. Иногда называют отрицательным флагом, так как значение 1 указывает на отрицательный знак числа.

Флаг нуля. Флаг, который равен 1, если результат последней операции равен 0, и 0 в противном случае.

Флаг отрицательный. См. Флаг знака.

Флаг переноса. Флаг, который равен 1, если при последней операции был перенос из старшего по значению разряда, и 0, если переноса не было.

Флаг прерывания. Разряд в секции ввода-вывода ЭВМ, который установлен, когда возникает событие, требующее от центрального процессора обслуживания. К типичным событиям такого рода относятся активность передачи на линии управления и исчерпание счетчика таймера.

Формат зонный десятичный. Двоично-десятичный формат, при котором каждый байт содержит одну десятичную цифру.

Формат упакованный десятичный. Двоично десятичный формат, при котором каждый байт содержит две десятичные цифры.

Фиксатор. Устройство, сохраняющее свое содержимое, пока в него не будут введены новые данные.

Фронт отрицательный (в двоичном импульсе). Переход от 1 к 0.

Фронт положительный (двоичного импульса). Переход от 0 к 1.

Функция ИСКЛЮЧАЮЩЕЕ ИЛИ. Логическая функция, которая истинна, если истинно любое из ее входных значений, но не оба сразу. Таким образом, эта функция истинна, когда ее входные значения не равны (т. е. если одно из них равно логической 1, а другое — логическому 0).

Функция Знак. Функция, равная 0, если ее параметр положительный, и 1, если отрицательный.

Функции работы со строками. Процедуры, позволяющие программисту работать с данными, содержащими символы, а не числа. Типичными функциями являются вставка, удаление, объединение, поиск и замена.

Ц

Цикл бесконечный, или *команда перехода на саму себя.* Команда, передающая управление на саму себя и выполняемая, таким образом, бесконечно (или пока ее не прервет аппаратный сигнал).

Цикл команды. Процесс выборки, декодирования и выполнения команды.

Ч

Часы реального времени. Устройство, прерывающее работу центрального процессора через регулярные интервалы времени.

Число без знака. Число, все разряды которого используются для представления его значения.

Число со знаком. Число, в котором один или несколько разрядов указывают на его попожительность или отрицательность. В обычном формате старший по значению разряд характеризует знак (0 – попожительный, 1 – отрипательный).

Ш

Шестнадцатеричная система счисления. Система счисления с основанием 16. В данной системе за десятичными цифрами от 0 до 9 следуют буквы от A до F (представляющие десятичные числа от 10 до 15).

Э

Эхо. Отражение на терминале переданной информации или возвращение на терминал информации, полученной с него.

Я

Язык ассемблера. Язык ЭВМ, в котором программист может использовать мнемонические коды операций, метки и имена.

Язык высокого уровня. Язык программирования, который нацелен на решение задач, а не на удобство преобразования в машинные команды. Компилятор или интерпретатор транслируют программу, написанную на языке высокого уровня, в программу, которую может выполнять ЭВМ. К распространенным языкам высокого уровня относятся Ада, Бейсик, Си, Кобол, Фортран, Паскаль.

Язык машинный. Язык программирования, который может восприниматься ЭВМ прямо без какой-либо трансляции лишь с числовыми преобразованиями.

Язык низкого уровня. Язык ЭВМ, в котором каждый оператор транслируется в одну команду на машинном языке.

СПИСОК ЛИТЕРАТУРЫ

1. Fischer W. P. Microprocessor Assembly Language Draft Standart. – IEEE Computer, December 1979, pp. 96 – 109; IEEE Computer, April 1980, pp. 79 – 80; IEEE Computer, May 1981, pp. 8 – 9.
2. Duncan F. G. Level-Independent Notation for Microcomputer Programs. – IEEE Micro, May 1981, pp. 47 – 56.
3. Osborne A. An Introduction to Microcomputers: Vol. 1. – Basic Concepts, 2-nd ed. – Berkeley, Calif.: Osborne/McGraw-Hill, 1980.
4. Shankar K. S. Data Structures, Types and Abstractions. – IEEE Computer, April 1980, pp. 67 – 77.
5. Tenenbaum A., M. Augenstein. Data Structures Using Pascal. – Englewood Cliffs, N. J.: Prentice-Hall, 1981. Этими же авторами написано несколько аналогичных книг для других языков программирования и ЭВМ.
6. Weller W. J., et al. Practical Microcomputer Programming: The Intel 8080. – Evanston, Ill.: Northern Technology Books, 1976.
7. Leventhal L. A., W, C, Walsh. Microcomputer Experimentation with the Intel SDK-85. – Englewood Cliffs, N. J.: Prantice-Hall, 1980.
8. Dollhoff T. Microprocessor Software: How to Optimize Timing and Memory Usage. Part One: Techniques for the Intel 8080 and Motorola 6800. – Digital Design, November 1976, pp. 56 – 69.
9. Seim T. A. Numerical Interpolation for Microprocessor-Based Systems. – Computer Design, February 1978, pp. 111 – 116.
10. Abramovich A., T. R. Crawford. An Interpolating Algorithm for Control Applications on Microprocessors. – Proceedings of the 1978 Conference on Industrial Applications of Microprocessors, Philadelphia, Penn., pp. 195 – 201. The proceedings are available from IEEE, 445 Hoes Lane, Piscataway, N. J. 08854.

11. Distler R. J., M. A. Shaver. Trial Implementation Reveals Errors in IEEE Standard. – IEEE Computer, July 1982, pp. 76 – 77.
12. Leventhal L. A. Z80 Assembly Language Programming. – Berkeley, Calif.: Osborne/McGraw-Hill, 1979, pp. 3 – 164 through 3 – 169.
13. Taylor C. L. Data-Block Transfer Program Is Efficient and Flexible. – Electronics, June 21, 1979, p. 147. [Имеется перевод: Тейлор К. Л. Эффективная универсальная программа блочной передачи данных. – Электроника, 1979, т. 52, № 13, с. 77, 78.]
14. Leventhal L. A. Take Advantage of 8080 and 6800 Data-Manipulation Capabilities. – Electronic Design, April 12, 1977, pp. 90 – 97.
15. Caplan G. Special Techniques Handle Critical Sections. – EDN, March 5, 1980, p. 90.

СПИСОК РАБОТ, ОПУБЛИКОВАННЫХ ИЗДАТЕЛЬСТВОМ McGraw-Hill

An Introduction to Microcomputers: Volume 0 — The Beginner's Book, 3rd Edition
An Introduction to Microcomputers: Volume 1 — Basic Concepts, 2nd Edition
Osborne 4 & 8-Bit Microprocessor Handbook
Osborne 16-Bit Microprocessor Handbook
8089 I/O Processor Handbook
CRT Controller Handbook
68000 Microprocessor Handbook
8080A/8085 Assembly Language Programming
6800 Assembly Language Programming
Z80® Assembly Language Programming
6502 Assembly Language Programming
Z8000® Assembly Language Programming
6809 Assembly Language Programming
Running Wild — The Next Industrial Revolution
The 8086 Book
PET®/CBM™ and the IEEE 488 Bus (GPIB)
PET® Personal Computer Guide
CBM™ Professional Computer Guide
Business System Buyer's Guide
Osborne CP/M® User Guide, 2nd Edition
Apple II® User's Guide
Microprocessors for Measurement and Control
Some Common BASIC Programs
Some Common BASIC Programs — Atari® Edition
Some Common BASIC Programs — TRS-80™ Level II Edition
Some Common BASIC Programs — Apple II® Edition
Some Common BASIC Programs — IBM® Personal Computer Edition
Some Common Pascal Programs
Practical BASIC Programs
Practical BASIC Programs — TRS-80™ Level II Edition
Practical BASIC Programs — Apple II® Edition
Practical BASIC Programs — IBM® Personal Computer Edition
Practical Pascal Programs
CBASIC
CBASIC™ User Guide
Science and Engineering Programs — Apple II® Edition
Interfacing to S-100/IEEE 696 Microcomputers
A User Guide to the UNIX® System
PET® Fun and Games

Trade Secrets: How to Protect Your Ideas and Assets
Assembly Language Programming for the Apple II™
VisiCalc®: Home and Office Companion
Discover FORTH
6502 Assembly Language Subroutines
Your ATARI™ Computer
The HP-IL System
Wordstar® Made Easy, 2nd Edition
Armchair BASIC
Data Base Management Systems
The HHC™ User Guide
VIC 20™ User Guide
Your IBM® PC: A Guide to the IBM® Personal Computer
Z80® Assembly Language Subroutines

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

– – переполнение 30, 31, 94 – 96